Available online at www.sciencedirect.com
ScienceDirect
Science of Computer Programming 71 (2008) 17–56
Science of Computer Programming
www.elsevier.com/locate/scico

Quantifying the yield of risk-bearing IT-portfolios

R.J. Peters, C. Verhoef*

VU Amsterdam, Department of Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

Received 23 March 2007; received in revised form 14 November 2007; accepted 14 November 2007
Available online 23 November 2007

Abstract

We proposed a method to quantify the yield of an IT-investment portfolio in an environment of uncertainty and risk. For various common implementation scenarios such as growing demands during implementation without deadline extensions we showed how to monetize their impact on the net present value. Depending on the business case this can lead to higher or lower gains. We also took failure of projects within an IT-investment portfolio into account, by appraising the loss in case of failure, resulting in a more realistic yield. To provide maximal insight into this yield, we proposed to treat it as a stochastic variable. We explained how to infer various portfolio yield distributions: discrete, continuous, and cumulative distributions, leading to useful summaries such as box plots and histograms. We argued that these information-rich characterizations support decision makers in taking calculated risks, and provided insight in how to address IT-specific risks and what such risk mitigation may cost. We explained our approach by quantifying the expected yield of a small four project portfolio under uncertainty and risk, and we provided the results for a larger and realistic IT-investment portfolio.

Keywords: Appraisal; IT-investment portfolio; Risk-bearing IT-portfolio; Uncertainty and risk; Scenario analysis; IT-portfolio management; Valuation; Discounted cash-flow (DCF); Net present value (NPV); Gain; Loss; Yield; Yield distribution; Portfolio yield; Internal rate of return (IRR); Pay back period (PBP); Return on investment (ROI); Risk-adjusted return on capital (RAROC); Weighted average cost of capital (WACC); Weighted average cost of information technology (WACIT); IT-investment management; Quantitative IT-portfolio management; Requirements creep; Time compression; Failure risk; Cost over-run; Time over-runs

1. Introduction

IT-portfolio management is concerned with the problem of managing the business value of the IT-investment portfolio. Benchmarking research companies like META Group [15] indicate that IT-portfolio management is the only method by which organizations can manage IT from an investment perspective—in line with the current state of business, and with a continuing focus on business value, risk, and costs. Usually an organization can choose from various alternatives and is searching for the portfolio whose yield is as high as possible while keeping the risk at an acceptable level. In other words, the most desirable portfolio is the outcome of a trade-off between risk and return.

A well-known measure to judge the business value of any investment is the economic indicator called net present value (NPV, abbreviated). The calculation of the net present value is based on the assessment of the future cash-inflows

* Corresponding author.
E-mail addresses: rjpeters@cs.vu.nl (R.J. Peters), x@cs.vu.nl (C. Verhoef).

0167-6423/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2007.11.001

and cash-outflows that can be attributed to the investments. Cash-inflows and cash-outflows in the far future are valued much lower at the present time than money that is earned or spent in the near future. The money is discounted against a fixed discount rate. Below we give the formula to calculate the net present value of an investment proposal.

$$\mathrm{NPV} = \sum_{t=0}^{n} \frac{C_t}{(1+r)^t}, \tag{1}$$

where n stands for the lifetime of the investment measured in years, t indicates the year of deployment of the investment, r is the annual discount rate, and C_t is the net cash-flow in year t.

1.1. Uncertainty in business value

In calculating the value of an IT-investment portfolio one should take uncertainty and risk into account. Of course there is the risk of receiving less benefit from the investment than estimated. We refer to this risk as the business domain risk. In case of a business-enabled IT-investment the business domain risk is the risk of making less profit from sales on the market place than expected. Besides the business domain risk there are also IT-risks. In calculating the added value we should take into account the chance of failure of the project, the possibility of budget over-run, the possibility of time over-run, and more. The cost consequences of the materialization of the risk of IT-project failure can be substantial and can alter the yield drastically. Benchmark data shows that IT-project failure is very common and a 30% failure chance is not unusual [16–18,23,33] (but see also Formula (7), expressing the risk of failure in this paper). In this paper we show how the chance of IT-project failure impacts the yield, and how it can be quantified and made visible in commonly seen development scenarios. We consider three often occurring scenarios where IT-risks play a substantial role.

- A rigid scenario where no change in functionality, no extension of deadlines, and no additional budget are allowed. We often see this rigidity in outsourcing deals where fixed-price, fixed-time, and zero-change policies are implemented.
- The scenario of cost over-run due to the phenomenon of requirements creep. Benchmark data show that requirements creep occurs often [23]. Due to growing demands both the total development costs and the operational costs increase.
- The scenario of cost over-run as a consequence of the phenomenon time compression in the presence of growing demands. Extending the scope while keeping to the original schedule is an often seen phenomenon in the IT-world. Ed Yourdon devoted a book to it called *Death march*—a survival guide to manage such mission-impossible projects [38].

For the three scenarios we know that they do happen, but we cannot attach a probability to it. Therefore, we must calculate their added value for each case so that a decision under uncertainty is accompanied by the impact in case some scenario materializes. Having done this, we will incorporate the chance of IT-project failure in them so that decision making under risk ensues. The decision maker is then provided with a large body of statistical material through which a rational decision can be reached. While it is possible in theory to prevent requirements creep and time compression, in practice you will always have projects subject to such uncertainties for instance by changes in the environment in which the systems are intended to operate. Let us explain. Suppose your idea was to launch a new software-intensive service, but you learn that the competitor will do so earlier than your nominal schedule to get the software operational. In case the first-mover advantage is important your project turns out to be subject to time compression. Or, you planned a system and while constructing it, suddenly the legislator announces new regulations. Then your project is subject to requirements volatility.

Of course, you can also calculate other scenarios to account for specific IT-risks and uncertainties, or different combinations. We took the above three scenarios since they are common, and serve the purpose to illustrate our approach. Requirements creep is the addition of requirements during the software construction phase, after the phase of requirements specification has been finalized. This is so endemic in software construction that there are benchmarks per industry for monthly growth rates [23]. However, these creep rates are not connected to useful versus useless volatility. In this paper we deal with two different requirements creep possibilities. One possibility assumes that the requirements creep does not add significant additional business value to the planned IT-investments. The consequence

of adding new requirements simply is that a larger IT-investment is going to be delivered than originally planned. Deploying the new system does not give rise to higher benefits. The other possibility assumes that adding new requirements also results in added business value. With the aid of the extended system more benefits can be generated, for example because a larger market share can be captured.

We speak of time compression when less time is available to do the job than is required from a technical viewpoint. Then the most appropriate production manner cannot be executed. The system must be built in a forced way as a result of which the total cost of development can surge to prohibitively high levels [31,30].

1.2. The expected value

For the risk of failure, we do not calculate different scenarios, but we incorporate this risk in the appraisals of our three scenarios. This is possible since we have failure distributions of the risk of failure. First we explain the basics. Roughly, the expected value $E(x)$ of a random variable x is the sum of the probability of each possible outcome of an experiment $p(x)$ multiplied by its value: $\sum x \cdot p(x)$. For instance, take a game with four chips in a bowl of which three are labeled 1, and one is labeled 2. You earn in dollars what is on the chip-label. So there is a chance of 3/4 on a single dollar and of 25% on two dollars, this is a two-point probability distribution. The expected value is $1 \cdot 3/4 + 2 \cdot 1/4 = 5/4$, which is what a player can expect to get on average in the long run [19]. If the organizers of the game asks a dollar a game, they will end up broke, for $1.25 they will play a zero-sum game, and for $1.50 the game will be paying off in the long run.

Likewise when the yield is subject to various outcomes with various chances, we can calculate its expected value. In our case, we speak of the *expected* yield, when we take into account the chance of failure of IT-projects. The adjective *expected* refers to the mathematical expected value of the outcome of the portfolio yield. So also here, the expected yield denoted $E(Y)$ equals the sum of its probability multiplied by its value. The probability of each outcome is described by an empirical probability distribution: the portfolio yield distribution. In this paper we will show how one can visualize the yield distribution of an IT-portfolio. The probability distribution as a whole provides much more information to decision makers than just the mean of all possible outcomes. Depending on their attitude towards risk a well-considered, fact-based decision about the acceptance of an IT-investment portfolio can be made.

1.3. Scope of the paper

In this paper we consider tailor-made, IT-enabled business investment proposals. We prefer to use the phrase *IT-investments* rather from software investments, as the investments are justified by business cases. We speak of an IT-enabled business investment if at least 25% of the total budget is spent on IT-activities.

In this paper we present a method by which the probability distribution of the yield of an IT-investment portfolio can be calculated and depicted in a graph. To demonstrate our computational framework, it is applied to an anonymized case of traditional, in-house system development (e.g. waterfall, spiral method). We assume that limited information is present, so that important key performance indicators like project duration, programmer productivity, and operational lifetime of an IT-system must be derived from benchmarks. To estimate the total cost of development, the yearly cost of operation of a system after delivery, and the chance of project failure we use formulas that have been estimated from public benchmark data concerning IT-projects. Also for outsourced development, formulas estimated from benchmark data can be used to get an estimate of productivity, cost, duration, and chance of failure [33,36]. This puts a CMM level 1 organization, which has no substantial IT-related historical data of its own, in the position to make a jump-start in analyzing its investment portfolio. If an organization has its own historical database it can use that data to estimate formulas that are more tailored to the organization, as generally the public benchmark data is less informative than company-specific information. No matter what kind of formulas you use, this does not affect our approach, only the input is more geared towards a specific organization.

Also, if an organization employs a different development method, e.g. it uses a component-based software development (CBSD) method [3,11], our approach is just as applicable. Using a CBSD-method generally implies a shorter development time and lower cost of development, if reuse of components is possible. Also the use of commercial-off-the-shelf components (COTS) often leads to significant benefits. The formulas to estimate the development cost, the yearly cost of operation and the chance of project failure will be different from the benchmark formulas we use for illustration purposes, but the computational framework and the interpretation of the results remain the same. It is beyond the scope of this study to go into the details of portfolios that contain also COTS packages.

There are experience repositories available for project managers to support them in making estimates of the amount of time it will take to integrate COTS components into a system, based on estimates of the amount of glue-code needed, and to quantify the risk of failure. The capabilities of these repositories are growing as more practitioners share there experiences with each other. A good example is the COTS Lessons Learned Repository (CLLR). The CLLR is a living repository that is used by many practitioners and can be accessed over the web. For more details on the subject we refer the interested reader to [12]. We also refer to the COCOMO 2.0 model: the new version of the Constructive Cost Model (COCOMO) for software effort, cost, and schedule estimation. COCOMO 2 also covers component-based and reuse-driven software development approaches [7,6]. The COCOMO 2 Post-Architecture model covers both the development and maintenance of a software product.

In dealing with IT-risks we limit ourselves to the risk of project failure. By this we mean the risk that no usable information system is delivered. The development project is stopped and no system is delivered at all or the system delivered is of no use for any business purposes. In both cases the calculated NPV through the business case cannot be realized. Of course one can also take into account the risk of budget over-run, time over-run, and/or the risk of solution underdelivery. For the sake of simplicity we limit ourselves to the case of project failure for explanatory purposes.

Further, we use four anonymized example projects each of a different size and with different business characteristics. We elaborately discuss these four example projects each displaying a unique aspect of our approach. We provide full calculational details so that others can reproduce our results. As a matter of fact as soon as this paper was disclosed on the Internet, we immediately obtained feedback from industrial people that scrutinized our results while revealing calculation errors. Thanks to our extensive treatise, others were able to fully appreciate our results, while improving them at the same time.

Organization of the paper. The rest of this paper is organized as follows. We start in Section 2 with some basic information on certainty, risk and uncertainty, and a discussion on intermediate scenarios. Then in Section 3 we provide background information on our experience with IT-portfolio modeling, and we will explain that also in the presence of erroneous and partial data it is already possible to reveal important patterns, useful to support decision makers. After these comforting insights, we review a number of basic benchmarks in Section 4. These benchmarks are necessary to engage in quantifying important IT-risks, which is what we treat in Section 5. Then in Section 6, we introduce a small four-project IT-investment portfolio and estimate its relevant key performance indicators (KPIs) using our basic benchmarks. With this information we are in a position to determine the gain of the individual projects for three situations: a rigid scenario, the case of growing demand, and the case of time compression in the presence of uncontrolled growth. The rigid scenario stands for a situation where no changes to the functionality, budget and duration of are tolerated. The outcome of these calculations is input to an appraisal of the entire sample portfolio. To that end we account for failing projects in the portfolio and calculate in Section 7 their yield and lift the results to the portfolio level. We do this in terms of a probability density and cumulative distribution. Each possible value for the portfolio yield is accompanied with its probability, giving the decision maker precise information on the feasibility of realizing that yield. After we illustrated all steps in the process, we apply it to an anonymized realistic IT-investment portfolio and provide the relevant distributions of the portfolio yield for all our scenarios in Section 9. In Section 10 we compare our valuations to an alternative appraisal method that we discussed in an earlier paper that concerned the valuation of single project IT-investments. Finally, in Section 11, we conclude the paper.

2. Certainty, risk and uncertainty

In statistical decision theory and practice it is common to make a distinction between certainty, risk and uncertainty [8,4]. If each possible action leads to an outcome which is known with certainty, decision making simply consists of selecting the most desirable one. Of course decision makers should be able to express their preference in the form of one or more objective functions. In the last case one speaks of multi-criteria decision making.

If each of the possible actions lead to a set of outcomes with known probability the decision is made in an environment of risk. Also in that case a decision maker can apply some algorithm to determine the action which is most desirable for the organization, if executives are able to specify their attitude towards risk in an appropriate mathematical form. For example, a decision maker can search for the action with the highest expected yield or the action with the lowest spread of yield. Or an executive can search for the action to which applies that the probability of exceeding some minimal yield level is maximal. Incidentally, this implies that the optimal value is only optimal with respect to some decision criterion. So there is no optimal value in an absolute sense.

If decision making is concerned with a number of possible scenarios that can actualize with unknown probability, decisions are made in an environment of uncertainty. No doubt, decision making under uncertainty is the most complicated one, as important information, such as the probabilities to attach to the different scenarios, is lacking.

In this paper we deal with a mix of decision making under certainty, risk and uncertainty, which is why we briefly discuss them to prevent confusion. The decision problem concerns the acceptance or rejection of an IT-investment portfolio proposal. There is uncertainty about the actualization of the various scenarios under which the individual IT-projects will be carried out. As already discussed, we consider some scenarios that frequently occur in practice. We cannot attach a probability to the occurrence to each of the scenarios, except the failure scenario for which we know a distribution. The other scenarios can happen, but with unknown probability. The organization therefore faces a problem of decision making in an environment of uncertainty for those scenarios.

As said before a well-known measure to judge the business value of any investment is the economic indicator called net present value NPV. However, there are other approaches to justify an IT-investment proposal. For example, the Real Option Valuation (ROV) approach is sometimes used to justify (large) investments in IT-infrastructure. The basic idea of the ROV-method is that the investment is split up in a small initial investment, which always should be made, and possible follow-up investments. Putting money in the initial small investment implies buying a so-called *call option* to undertake follow-up investments in the future. There is no obligation to cash the call option, however. If one decides not to undertake the possible follow-up investments, because additional information has become available which makes clear that it is unattractive to do so, the maximal loss of the investment equals to the cost of the small initial investment. It is worth to note that the way we distinguish between risk and uncertainty is not inconsistent with the way real options analysis accounts for risk and uncertainty. If at time t_0 no probabilities can be attached to the actual realizations of the scenarios under which the follow-up investments are profitable we have to do with decision making under uncertainty. The decision maker is then provided with information about the return on investment under different scenarios that can actually realize and a decision depends on the organization's attitude toward taking risk. If the degree of uncertainty is known, you can compute the probability distribution of the real option value and decide whether or not that probability distribution is acceptable to the organization. However, it is remarkable that the ROV-approach does not account for the possibility that the initial investment to obtain the real option fails; we do take that very real option into account by quantifying the loss exposure due to failing projects.

In decision theory, gains, losses or both always appear explicitly in the analyses. In our case the gain is the NPV that is received if the IT-project is implemented successfully. The loss of an IT-project is the amount of money lost if the project fails. That amount equals to the tax-break corrected, discounted development costs. The decision problem is thus formulated in terms of scenarios that can actualize. The business case for an IT-investment proposal is assumed to be known with certainty and depends on the scenario in question. In case of a scenario under which there is requirements creep that has to be processed there are two possible courses of action between which the decision maker can choose. Suppose that the decision maker can accept the project time over-run due to processing the additional requirements that are submitted, or can decide to speed up system delivery by time compression. Then we are dealing with decision making under certainty. Depending on the business, some course of action is more optimal than another and it is simply a matter of calculation to determine which action that is.

Under each scenario, there is the risk of failure of the IT-investments. We modeled the chance of failure as a function of the project size and, thus, a probability can be calculated as such. This implies that we are dealing with decision making under risk.

In the following we present a method to quantify the risk of receiving less profit than some fixed amount from the IT-investment portfolio. To be more precise, we present a method by which the discrete density and the cumulative probability distribution function of the portfolio yield can be calculated and depicted in a graph.

By deriving portfolio yield distributions for all scenarios that can actualize, we provide the decision maker with valuable quantitative information to take decisions in an environment of risk and uncertainty. In this paper we will not discuss criterion setting for decision making. This depends on the organization's risk appetite.

2.1. Discussion

In fact we consider typical but also extreme scenarios. For the rigid scenario characterized by zero-change policies, we demand that all IT-projects must be carried out in accordance with the planning. And for the scenario under which additional requirements are allowed during the construction phase, we uniformly apply that to all projects. The reason

is that we often see these phenomena, and in fact they can be spotted in exploratory data analyses, as we will see in the next section, or alternatively in paper [37].

Clearly, there are many in-between scenarios. For each possible scenario, which has a non-negligible probability of occurrence, the cumulative distribution function of the portfolio yield can be calculated and depicted in a graph. Our experience is that when fine-grained scenarios are taken into account to optimize for portfolio value, this is inevitably conjoined with a mature IT-governance structure that allows for flexible portfolio, program, and project management of the IT-function. So in an IT-investment-aware organization, IT-governors have more decision making possibilities than just accepting or rejecting a portfolio.

Although the probability of occurrence of each scenario is unknown, this does not imply that the organization cannot influence the probabilities. It is not just a case of force majeure. By taking appropriate measures the organization can decrease the probability of occurrence of unwanted scenarios with a low yield. For instance, preventable requirements creep. Also, by investing more resources in professional project management, the organization can decrease the chance of project failure. This clearly depends on the time and costs whether or not it is wise to invest money in taking such measures. In [34], we calculated that already from a cost-point of view such (often large) investments in professionalizing the IT-function and the business interacting with it can pay off. More on our experience and IT-governance maturity in the next section.

3. IT-portfolio modeling

Both authors have decades of industrial experience with the quantitative side of the IT-function to support decision making. The first author set up quantitative thinking within a large international bank-assurance conglomerate, and the second author advised many organizations on financial/economical matters concerning information technology. Building investment-specific IT-portfolio models is part of such work, and during that work both authors met and started cooperating. Since a few years the authors have permission to share some of their results with others.

Scientific publications on quantitative IT-portfolio management are still sporadic and recent. In the journal *Science of Computer Programming* the second author surveys the basics of quantitative IT-portfolio management [33]. Since then the area attracted industrial and academic attention, sparked off research projects, an IEEE Conference, and inspired others in their writings [24,5]. Our work basically consists of developing mathematical and statistical models that enable executives of organizations to make a fact-based decision re the use of one of their largest production factors: information technology. Such models consist of mathematical equations that specify the relationship between IT-variables and non-IT-variables. The parameters are statistically estimated from in-house data or suited benchmark data. For example, a formula can specify the relationship between the size of a system to be built and the average productivity a system developer can reach. Other examples are formulas calculating the total cost of a development project or formulas calculating the chance of failure of an IT-project. It is of utmost importance for decision makers to obtain quantitative information about the chance of failure of an ambitious IT-project before starting it. If the risk is unacceptable, management can consider taking appropriate measures to mitigate the risk. Of course the costs of such measures must be within an acceptable range, and that too can be quantified in some cases.

Preferably, the parameters of the equations should be estimated from in-house data. However, in many cases sufficient data is lacking and/or no reliable data is available. In terms of the Capability Maturity Model [28] (CMM for short) such organizations have a lower maturity level than two. On the CMM scale a score of one corresponds to the level of a starter that has not yet its development processes in place. Because no measurements are carried out in a systematic way, no historical data regarding productivity and quality have been captured over time. CMM level five implies that that the organization is a professional one and is continuously further optimizing the quality and productivity of its software process development. Organizations at CMM level 1 can resort to public benchmark data as a surrogate, and this case is elaborated on in [33]. However, one should be cautious, and use the quantifications as order of magnitude estimates, which is still better than gut feel. The reason is that the current data from public databases do not fit well to all organizations. In that case one should try to select the data of peers from public benchmark databases.

3.1. Raw diamonds of data

Our research is focussed on the development of quantitative tools towards managing the IT-portfolio. It often seems that an organization has no IT-related data at all. During our research it turned out, however, that many companies

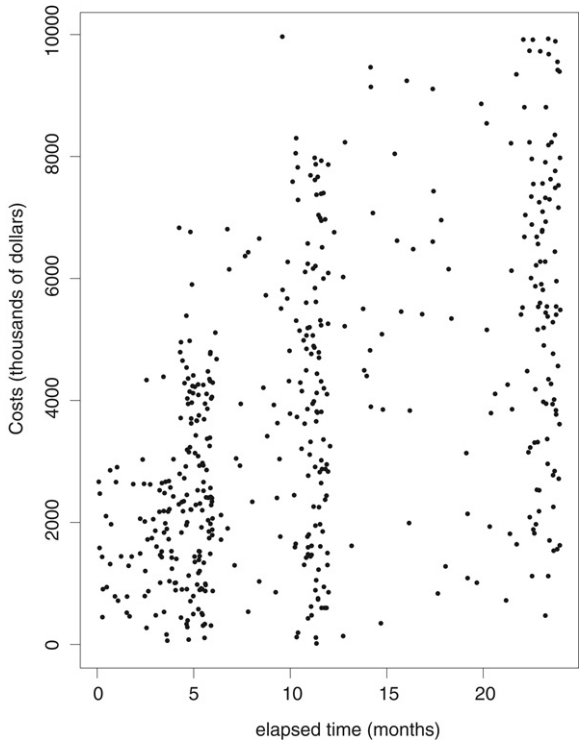

Fig. 1. Cost–duration view of a sample IT-portfolio containing 495 projects, at a total cost of 1.8 billion dollars.

are in possession of a lot of unstructured and scattered historical data regarding the production factor IT. The data is usually spread over various sources. For example, the general ledger system may be a source of information regarding the cost of maintenance and exploitation of systems over time. Also much relevant information may be extracted from the source code of systems in use. All data has to be collected from the various sources and related to each other in an IT-portfolio database.

Practice has shown that many organizations are unaware of the fact that they are in possession of a gold mine of data. Even with relatively sparse and partly incorrect data it often is possible to learn from it and detect patterns that otherwise would go unnoticed. In Fig. 1, we give a typical example of pattern recognition which is helpful in assessing the impact of IT-governance rules (it is taken from [37]).

The example is derived from a real-world case. The scatter plot in Fig. 1 shows the relationship between the total cost of the development and the duration time for a large number of IT-projects, costing two billion dollars. The scatter plot has been generated with a statistical package. A scatter plot just plots two numeric objects in a single graph. What strikes in Fig. 1, is the column structure of the data. At the time stamps 6 months, 12 months and 24 months one can observe a large dispersion in the development costs of the corresponding systems. This pattern is highly counterintuitive. One would expect higher development costs for projects with longer duration times. Since the logical relationship is lacking many people are tempted to conclude that the data are unreliable or too many data are missing. However, this conclusion is false: this is a diamond in the raw. There is a good reason why the relationship is not to be found. The cause resides in the specific IT-governance rules of the company. For administrative reasons budget allocation is fixed to regular periods of 6, 12 and 24 months. However, the optimal manner of production does not fit into this administrative rule. If the most appropriate production manner requires 7 or 8 months time, the development costs may increase dramatically if the system is forced to be built within a shorter time frame. The same amount of work has to be done within a shorter time frame, which implies doing work in overtime, appointing more staff to do the job, and not being able to take the proper time for the work. The consequence of more staff is also that more effort is needed to coordinating the work. So there are a large number of systems in the 6 months column that are built under pressure and therefore have higher costs than would have been necessary under normal circumstances. The same line of reasoning applies to the 12 months column and 24 months column projects.

Without the above simple explorative data analysis this pattern would not have been revealed. This example makes clear that the data do not have to be perfect to be able to detect patterns in it, but can still enable management to improve the governance rules and save a considerable amount of money, since unnecessary time compression that plagued this portfolio was diagnosed and mitigated. Later on we will extensively discuss time compression. For now it suffices to know that already with simple means quantitative IT-portfolio management can be jump-started. For publications testifying of a number of results like the one we alluded at we refer the interested reader to [33,36,35,37,34].

In this paper we will show how the impact of IT-risk exposure on the calculation of the yield of an IT-investment portfolio can be quantified, given the projected cash-flows from the investments. But before we commence with that, we discuss some basic benchmarks.

4. Basic benchmarks

As discussed in Section 1 we assume that limited information is present, so that important key performance indicators like project duration, programmer productivity, and operational lifetime of an IT-system must be derived from benchmarks. We also limit ourselves to IT-investments that are tailor-made. Often when you have a positive business case with a software package, the costs of procurement and tailoring are such that a market advantage is limited. With tailor-made IT-systems this is less clear, if only since it is quite daunting to successfully produce correctly operating software-intensive solutions. Of course, the methods we lay out in this paper will also hold for other types of IT-investments, but then the formulas that we deploy for their cost, schedule and associated risks will be different. However, the appraisal methods to derive yield distributions at the portfolio level will not differ.

4.1. Function points

A function point is a synthetic measure expressing the amount of functionality of a software system (to be built). It is programming language independent, and is very well-suited for cost estimation, comparisons, benchmarking, and therefore also a suitable tool for developing quantitative methods to manage IT-portfolios [1,2,10,13]. Function points meet the acceptable criteria of being a normalized metric that can be used consistently and with an acceptable degree of accuracy [26,25,34]. Function points have proven to be widely accepted both by practitioners and academic researchers [21,20,22,23]. Function point counting is a widely used functional sizing measure of IT-projects and IT-systems. It is known that function points can usually be counted at a rate of hundred function points per hour, although depending on the clarity of the requirements this can drop to 30–40 function points per hour. There are different techniques known to carry out a function point analysis. The International Function Point Users Group provides an ISO standard for function point counting [14]. As said before, function points have proven to be the most reliable metric for measuring the size of a system. For a detailed discussion on assessing the potential bandwidth of function point counting the reader is referred to [34,36]. If an organization is after the most accurate counting it can use a combination of certified analysts and a combination of commercial tools, in-house developed tools, and custom statistical/mathematical analyses. For now it is important to know that there is some valid synthetic metric expressing the size of a software system that is suited for financial/economic analyses.

4.2. Estimation project duration

Benchmark data indicate a nonlinear correlation between the size of an IT-system to be built and the project duration. In our example calculations we use the following relation, estimated from Jones' public benchmark database [22, p. 202].

$$f^{0.39} = d. \tag{2}$$

Here f stands for the number of function points of software and d for project duration, measured in calendar months. Jones claims to have the largest knowledge base on software projects in the world with more than 10 000 projects, and growing. So despite inevitable inaccuracies in formula (2), they compactify an immense amount of experience data that is good enough for the order of magnitude estimates. Note that the schedule power varies depending on the industry you are in, or on the type of project you are executing. The schedule power 0.39 is specific for Management Information Systems (MIS) projects that are part of all businesses and omnipresent in the

financial services and insurance industry. For example, the schedule power specific for softwares subjected to military standards is 0.43. Since our example IT-projects are typical for the financial services and insurance industry, we use the power 0.39 in our calculations. Of course, if you have internal data, you can use that data, or an internal benchmark instead of formula (2). Then the numbers will be somewhat different, but the methods to appraise the portfolio will remain the same.

4.3. Estimating productivity

In addition to schedule power benchmarks, there are two other benchmarks, also taken from Jones' database on software projects [22, p. 202–203]. These are overall benchmarks, not specific for the MIS industry. The first benchmark formula applies to software development projects.

$$n = \frac{f}{150}. \tag{3}$$

Here n is the number of staffs needed for the project and f is the number of function points that are to be built. The constant 150 is specific for this type of work and is called an assignment scope. The formula tells us that on the average one developer can handle 150 function points. So if the software size of a system amounts to 600 function points, the development team is 4 staff members according to the benchmark. In our calculations to follow we use these benchmark figures as a proxy for the average productivity of in-house developers. Organizations with sufficient historical data can use their own average productivity figures instead.

From formulas (2) and (3) it follows that the productivity for IT-system development (measured in a number of function points per staff month) decreases sharply as the size of the system to be built increases. There are a number of reasons for this. An important one is that more effort has to be put into coordination between the members of the development team as the size of the system increases. Jones has found nonlinear productivity decrease of in-house developers if size increases in his database. In [23, p. 191], it is stated that for in-house developed MIS systems the average work hours per function point are 4.75 for 100 function point (FP) systems, 13.93 for 1000 FP systems, and 38.41 for systems of 10 000 function points. Or equivalently, 27.8, 9.46, and 3.44 function points per staff month, respectively. So, assuming nonlinear behavior, we can extrapolate and find the following relation (taken from [35]).

$$hfp(f) = 0.6390553 \cdot f^{0.4448014}. \tag{4}$$

In this equation *hfp* is short for hours per function point, and f is the amount of function points. Alternatively you can use roughly the inverse: function points per staff month, which is denoted *fpm*, and calculated as follows: $fpm = 20.5/hfp$, since on average there are 20.5 days in a staff month.

Another formula is useful to estimate the number of staff needed for keeping an application up and running (operational) after delivery. It is taken from Jones [22, p. 202–203].

$$n = \frac{f}{750}. \tag{5}$$

Here n is the number of staff needed to do the job and f is the number of function points to be maintained. For this type of work the benchmarked assignment scope is 750 function points. The formula tells us that on average one team member can handle 750 function points. So if software has a size of 750 function points a single staff member keeps it up and running.

4.4. Estimation software lifetime

From an investment point of view it is important to have an idea of the operational lifetime of an investment. Namely, to be able to say something about minimal total cost of ownership it is important to know how long the software will be operational. While constructing the business case of an IT-investment proposal one needs to estimate the lifetime of the system. If the lifetime cannot be estimated based on business arguments, one can resort to the following benchmark formula, taken from [33], and adapted from Jones' work.

$$y(d) = d^{0.641}. \tag{6}$$

In formula (6), y stands for the number of calendar years the software will be deployed and d for the development time duration, measured in calendar months. For example, for a 36 month development project the delivered software will be deployed $y(36) = 9.9$ years according to the benchmark formula. The rationale behind Eq. (6) is that it normally takes a longer time to write off a larger investment than a smaller one. To simplify matters we have taken formula (6) as a starting-point in Section 7 in making assumptions about the deployment time of the systems to be built.

4.5. A word of caution

We stress that the reader should not use these formulas as a sole decision means for individual software project contracting purposes, to decide on individual resources for large software projects, or to decide on upcoming individual projects with potential large business impact. For these applications the above-mentioned formulas are not specific enough, as they are estimated on the basis of public benchmark data that generally will not be representative for the own organization. Just compare to Fig. 1, where other patterns for cost–duration are found than you would expect from benchmark formulas. However, in portfolio calculations the variations tend to cancel each other out, and for such type of effort where go/no go decisions are typical the just introduced benchmark formulas are a useful tool to support decision making in addition to other more qualitative means aiding executives [33].

5. Important IT-risks

In developing an IT-portfolio an organization runs the risk of project failure. Also the organization faces the possibility that different scenarios can materialize with unknown probability. The cost consequences of the materialization of IT-risks and the scenarios opening to it can be substantial and can change the added value of the investment dramatically. We recall the three important scenarios that we will deal with.

- Rigid scenario.
- Cost over-run due to requirements creep.
- Cost over-run due to time compression with scope creep.

As said earlier, in a rigid scenario no changes to either functionality, time or budget are allowed. Requirements creep is the addition of requirements during the software construction phase, after the phase of requirements specification has been finalized. We speak of time compression when less time is available to do the job than is required from a technical viewpoint. Consequently, the most appropriate production manner cannot be executed. The system must be built in a forced way as a result of which the total cost of development can increase dramatically. In this section we first show how one can quantify the risk of failure of an MIS-project built in-house as a function of software size. Next, we show how one can quantify the impact of requirements creep and time compression on the project duration, the total cost of development and the yearly cost of operations during its operational life-time.

5.1. Risk of failure

For the risk of failure we already stated that we will use a different approach, since we know more about that risk than just that it happens. We used public benchmark data from Jones [23, p. 192], to estimate the relation between the chance of project failure and the size of the system to be built, measured in function points. Based on Jones' data points that represent in turn the data of many projects the following formula has been estimated [33].

$$cf(f) = 0.4805538 \cdot \left(1 - \exp\left(-0.007488905 \cdot f^{0.587375}\right)\right). \tag{7}$$

In formula (7), cf stands for the chance of failure and f for the number of function points. Note that the exponent of f is given with high precision. Also the other parameters are specified with the same kind of exactness. This may look a bit overdone, but that it is not the case at all. For example, rounding off the number to 0.5874 upwards would give a curve with a quite different shape than the curve presented in Fig. 2. The graph shows that the chance of failure increases sharply as the size of the system increases. At a size of 6000 function points the chance of failure already equals to 34%.

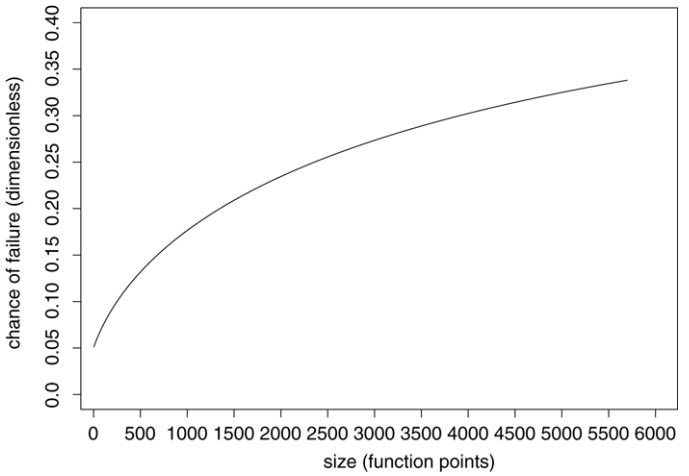

Fig. 2. Chance of failure as a function of project size.

Formula (7) should not be used for systems above 100 000 function points, as the asymptotic behavior of the formula prevents the chance of failure going to one when the size of software goes to infinity. For those cases another formula is to be fitted through the data, that has the advantage of being more accurate in the higher function point range, but the disadvantage of fitting less appropriate within lower ranges. Since our example projects are in the range that the formula covers, we can safely use it. Apart from that, the average size of MIS-systems is between 1000 and 2000 function points, so also for others it will often not be problematic to utilize formula (7).

5.2. The impact of time compression

The risk of budget over-run increases considerably if enough time is not available to apply the most appropriate production manner. This phenomenon is known as time compression. It is trying to do more work in a time frame than you would normally do from a technology viewpoint. The impact on the development cost can be estimated with the aid of the following relation between time and effort, which is taken from [31,30].

$$e \cdot d^{3.721} = constant. \tag{8}$$

Here e stands for effort, measured in person hours and d for the duration of the project in calendar months. This relation has been found empirically by father and son Putnam, and is used in their software cost estimation tools [31]. If the duration and effort needed to built a software system is known for, say the nominal condition we can calculate the *constant* from that. Subsequently, we can use the formula to estimate the effort if the duration of the development is less than nominal, thus the most appropriate production manner is no longer applied. By multiplying the increase in effort by the cost per working hour we can estimate the rise of the development costs due to time compression. It then becomes possible to make a trade-off between the benefits of time-to-market and higher development costs.

We give an example of the application of Formula (8). Let us assume that the size of a system to be built is 5565 function points. According to the benchmark Formula (2) the time needed to built the system under normal conditions (from a technology viewpoint) is $5565^{0.39} = 28.9$ months. The total cost of development is $17 863 851. If we assume that on average one developer can handle 150 function points the number of development staff is 5565/150 = 37.1. If we further assume that each developer works 200 days a year and a working day counts 8 hours, the effort in hours if the most appropriate production manner is applied equals to:

$$e_{\text{nom}} = (37.1 \cdot 200 \cdot 8)\left(\frac{28.9}{12}\right) = 142\,911,$$

where e_{nom} stands for the effort if the nominal production manner is applied. We calculate the constant in Eq. (8):

$$142\,911 \cdot 28.9^{3.721} = 38\,946\,516\,243.$$

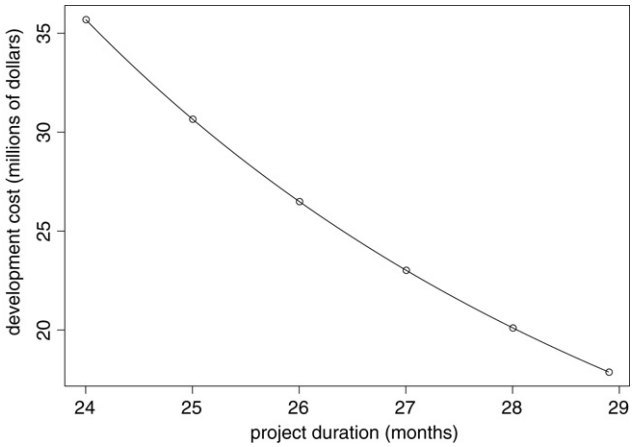

Fig. 3. Time–effort trade-off curve with various schedule compressions highlighted with an open circle, indicating strong cost increases for shorter time frames.

Now we have all the information needed to estimate the additional working hours required to implement the system earlier than within 28.9 months. For example, let us calculate the number of additional working hours needed if the deadline is changed to 24 months. We first calculate:

$$e_{24} = 38\,946\,516\,243 \cdot 24^{-3.721} = 284\,907,$$

where e_{24} stands for the number of working hours required to build the system if the project duration is shortened from 28.9 months to 24 months. The number of additional working hours equals to:

$$e_{24} - e_{\text{nom}} = 284\,907 - 142\,911 = 141\,896.$$

So, an additional 141 896 working hours are required just to make the deadline of 24 months. If we assume the cost per working hour to be \$125 the additional cost to make the deadline are:

$$(e_{24} - e_{\text{nom}}) \cdot 125 = 17\,749\,475$$

totaling to a cost of development of $17\,863\,851 + 17\,749\,475 = \$25\,613\,326$. In the same way we can calculate the amount with which the development costs increase if the time compression is 0.9, 1.9 and 2.9 months. The outcomes of the calculations are represented in the graph of Fig. 3.

The curve in Fig. 3 is project-specific. The constant in Formula (8) differs by case and has to be calculated each time again. Note that the costs also increase if much more time is spent on building the system than the nominal time of 28.9 months. To estimate the cost rise in that case another formula is needed. We refer the reader to [37] for details. Formula (8) holds for estimating the cost increase in the case of time compression.

5.3. Impact of requirements creep

Requirements creep is the addition of requirements during software construction, which is after the requirements were set. This is so common in software construction that there are benchmarks per industry for compound monthly growth rates. In this paper we deal with two different requirements creep scenarios. One scenario assumes that the requirements creep does not add extra value to the system to be delivered. The consequence of adding new requirements simply is that a larger system has to be delivered than originally planned. Deploying the new system does not give rise to higher benefits, but there are higher operational costs, and a loss due to longer development schedules. The other scenario assumes that adding new requirements results in improved cash-flows. With the aid of the extended system more benefits can be generated, for example because a larger market share can be captured.

In our calculations we assume a requirements creep of 2% per month, which was recommended by Jones if no other information is available. Of course, also industry-specific requirements volatility rates can be used, see e.g., Jones [23]. We just use the 2% to explain how to calculate requirements creep. Let f_{start} denote the size of the system

Table 1
Summary of the four example projects

#	f	d	tcd	y	yco	hfp
1	100	6	66 000	3.15	26 400	4.14
2	585	12	78 000	5	156 000	10.9
3	1036	15	1726 667	5.7	276 267	14.02
4	3460	24	9222 000	7.67	922 000	23.97

(in function points) at the start of the development project and f_{end} the software-size at the end. We denote the project duration, measured in months, by d. Now we can calculate f_{end} using the following formula.

$$f_{\text{end}} = 1.02^d \cdot f_{\text{start}}. \tag{9}$$

For example, let us consider a 100 function point software system. The estimated duration of the development project is 6 months according to benchmark formula (2). With a requirements creep of 2% per month the size of the software system at the end of the development period has grown to $f_{\text{end}} = 1.02^6 \cdot 100 = 113$. The new development duration can be estimated using formula (2) again and amounts to $d = f_{\text{end}}^{0.39} = 113^{0.39} = 6.3$ calendar months.

6. A small sample portfolio

Now that we covered the basics, we are in a position to present a simple example portfolio so that we can illustrate the machinations necessary to calculate the various investment scenarios for each individual project. With that, we are able to account for losses and correct the NPV for that so that a risk-bearing IT-investment portfolio yield can be derived.

We consider a portfolio which consists of four anonymized example IT-projects. We refer to these projects as project 1, project 2, project 3, and project 4. In order to explain our approach we hand-picked our example projects each of a different size. Project 1 must deliver a system of 100 FP, project 2 a system of 585 FP, project 3 a system of 1036 FP, and project 4 a system of 3460 function points.

In Table 1 the characteristics of our 4 example projects are summarized. Using Formula (2) the development duration d can be estimated for each of the example projects. To be able to estimate the total cost of development, abbreviated *tcd*, the following assumptions have been made. First, we assume the assignment scope of a system developer to be 150 function points, as in Formula (3). So, the size of the development team equals to $f/150$ FTE (full time equivalent). We further assume a daily burdened rate of $1000 both for development and maintenance, and 200 working days per year. The total development cost can then be calculated as follows.

$$tcd = \frac{f}{150} \cdot 200 \cdot 1000 \cdot \frac{d}{12}.$$

To estimate the yearly cost of keeping a system operational during its life-time we assume that one member of the maintenance team can handle 750 function points, as in Formula (5). In that case the maintenance team consists of $f/750$ FTE. With the earlier made assumptions, we can estimate the yearly cost of operation (*yco*) as follows.

$$yco = \frac{f}{750} \cdot 200 \cdot 1000.$$

The system's operational lifespan can be estimated with benchmark Formula (6). Using Formula (4) we calculated the productivity in hours per function point.

6.1. Scope creep

With requirements creep of 2% per month the size of a system to be built increases. By that the development duration, the total development cost and the yearly cost of operation will increase. The amount with which the size of the system increases due to requirements creep for our sample portfolio has been calculated with the aid of Formula (9). Table 2 shows the increase of development cost and yearly cost of operation due to requirements creep of 2% per month for the four-project portfolio. The proportional increase is given under the absolute increase. The

Table 2
Impact of requirements creep at 2% per month for the four-project portfolio

#	f_{start}	f_{end}	d	tcd	yco
1	100	113 13%	6.3 5%	78 974 19.7%	30 031 13.8%
2	585	742 26.8%	13.2 10%	1085 313 39.1%	197 846 26.8%
3	1036	1394 34.6%	16.8 12%	2602 133 50.7%	371 733 34.6%
4	3460	5565 60.8%	28.9 20.4%	17 863 851 93.7%	1484 051 61.0%

Table 3
Impact of time compression in addition to requirements creep at 2% per month for our small sample portfolio

#	f_{start}	h_c	h_{tc}	$\Delta\$$	tcd_{tc}
1	113	632	763	16 346	95 320 20.7%
2	742	8 683	12 259	447 034	1532 347 41.2%
3	1394	20 817	31 737	1 364 936	3967 069 65.6%
4	5565	142 911	284 907	17 749 475	35 613 326 99.4%

larger the system size the larger the proportional increase. Of course one can take measures to address requirements creep during the development process. But such measures will have their price and limitations: sometimes external effects force a project into a requirements creep scenario. Table 2 gives an indication of how much money you can afford to invest in mitigating the risk, or how much it is going to cost you if you cannot address this IT-risk. It is noteworthy to observe that development costs increases minimally 20% up to 94% for the largest project. Additional yearly operational costs-surge from 14% for the smallest system to 60% for the largest. So, unless there is a good business reason for requirements creep that justifies these soaring increases it is much better, much cheaper, and less risky to mitigate uncontrolled growth to more healthy volatility rates for instance under 1% for large projects.

6.2. Time compression

Table 3 shows the increase of the development cost if the longer project duration time due to requirements creep is restored to the originally planned schedule without requirements creep. This combination of IT-risks is often seen: a deadline is set, but the scope is extended along the way without actualizing the other KPIs. For project 1 the time compression is 0.3 months (from 6.3 back to 6); for project 2 the time compression is 1.1 months (from 13.1 back to 12); for project 3 the time compression is 1.8 months (back from 16.8 to 15), and for project 4 the time compression is 4.9 months (back from 28.9 to 24). The proportional increase of the development cost is also provided. With h_c we imply the hours of development effort in case of creep, and with h_{tc} we mean the hourly effort in case of time compression and requirements creep, further $\Delta\$$ stands for the cost increase due to time compression in the presence of requirements creep, tcd_{tc} is the total cost of development in case we have the compounded IT-risks of doing a project under time compression with scope creep.

The number of staff hours needed to built the system without time compression (column 3 in Table 3) has been determined by multiplying the number of staff needed by the number of working hours per year and the project duration measured in years. For example, for project 1 we find: $112.6/150 \cdot 6.3/12 \cdot 1600 = 632$ hours. The number of staff hours needed to built the system in case of time compression (column 4) has been calculated by using the formula of Putnam senior and junior [31], see Formula (8). We already illustrated earlier how one can estimate the number of staff hours additionally needed if the system has to be built under time compression. By multiplying this number

Table 4
Chance of failure as a function of project size for the nominal case, and the case with 2% requirements creep per month

#	f_{start}	cf	f_{end}	cf_c
1	100	0.0506	113	0.0544
2	585	0.1302	742	0.1464
3	1036	0.1717	1394	0.1967
4	3460	0.2847	5565	0.3339

by $125 (our assumed hourly compensation rate) the increase of the development cost due to time compression is obtained and reported on in column 5 of Table 3. Especially for the largest project in the portfolio we note that the cost increase of 60% due to requirements creep surged further to almost 100%. While this still can be a good business case given better cash-flows combined with a first-mover advantage, it is worthwhile to recalculate the projected NPV of this scenario, since a cost increase of one hundred percent will inevitably change the net cash-flows negatively if no substantial inflow is projected. Namely, the delivery of project 4 is 4.9 months earlier and therefore must gain more than $17 749 475 to compensate for the higher development cost. Indeed, the time compression of project 4 is high, about 17%, much higher than the time compression of 9.2% and 4.8% of projects 1 and 2. When we introduced time compression we used exactly the size of project 4 as an example, and we recall that its total cost of development already rises with $2.2 million at a time compression of 0.9 months.

6.3. Project failure

Table 4 summarizes the various chances of failure for the sample portfolio. Recall Formula (7) calculating the chance of failure of a project for a given size in function points. We provide two scenarios: the case where there is no requirements creep and the case where growing demand is not capped. The small project in the portfolio runs a chance of about 5% to fail, and the difference for the two scenarios is not that large. For project 4 the chance of failure is in the rigid scenario already 28.5% and this increases to 33% in case of scope extensions. The latter differences are fairly large and should be accounted for in the business case, which is exactly what we are going to do to appraise our entire portfolio of IT-investments.

7. Appraising the projects

In the preceding section we discussed the IT-characteristics of our small IT-investment portfolio consisting of four software projects. In this section we make assumptions about the earnings projected for each project during the years the software will be deployed. We will do this via the classical economic indicator net present value that we introduced earlier. In many NPV-calculations in practice a discount rate of 12% a year is used. We have done so in our calculations. The intended interpretation of this discount rate is that one assumes that there is an investment alternative, which yields an annual return of 12% of the invested amount. We note that for each organization a specific discount rate can be calculated or estimated via industry benchmarks. We just use 12% since it is quite common and to illustrate our approach.

7.1. A small sales system

We now dive into the business side of the small sample portfolio. We start with the first small project. In fact, this project serves to explain how we carry out our net present value calculations. Project 1 concerns the development of a relatively small system, which enables the sales of a new savings account product. The market analysts estimate the yearly earnings starting after the system has become operational at one hundred thousand dollars a year. The product lifetime is estimated as 4 years. Note that the estimated operational lifetime of a 100-function point system is about 3 years according to benchmark Formula (6). Subsequently, we will calculate the net present value for project 1 for three different cases: the rigid scenario, a scenario with 2% scope creep, and a scenario where despite scope creep no deadline extensions are allowed.

Rigid. Table 5 shows the NPV-calculation for the case that there are no changes to functionality, budget, or time. Since there is no time over-run the duration of the project is indeed six months. Let us explain the abbreviations:

Table 5
The NPV for project 1 carried out under a rigid regime

t	inv	ops	inc	gr C_t	C_t	DCF
1	66 000	13 200	50 000	−29 200	−20 440	−18 253
2	0	26 400	100 000	73 600	51 520	41 061
3	0	26 400	100 000	73 600	51 520	36 682
4	0	26 400	100 000	73 600	51 520	32 767
NPV						92 257

Table 6
NPV-calculation of project 1 with 2% requirements creep but no time compression, so with a project duration of 6.3 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	78 974	14 264	47 500	−45 738	−32 017	−28 591
2	0	30 030	100 000	69 970	48 979	39 036
3	0	30 030	100 000	69 970	48 979	34 873
4	0	30 030	100 000	69 970	48 979	31 151
NPV						76 469

t is as before in the NPV-formula (see Formula (1)) the year of deployment of the investment. Then inv is short for investment, which is divided into the years of investing. For this project there is only one year when money is invested. There will be operational costs, like maintenance, enhancement, and more; we abbreviate that with ops. The earnings or income is abbreviated inc. The gross cash-flow in year t, abbreviated gr C_t, consists of subtracting the investment and operational costs in that year from the income. For year 1, this amounts to a gross negative cash-flow of 29 200. The net cash-flow C_t in year t, is as in Formula (1) the net cash-flow. It is obtained by deducting 30% tax each year. Of course, each country has a different tax system, and therefore this column can differ on a state-by-state or country-by-country basis. Then again, we wish to illustrate how to appraise IT-portfolios and so we just took some tax-regime: deducting 30% each year of operation. Last, DCF is short-hand for discounted cash-flow. In fact, it consists of the outcome in year t of the net cash-flow divided by a so-called *discount factor* $(1 + r)^t$, with *discount rate* $r = 0.12$, or 12%. These discount factors are summarized in Table 25 for various rates, among which our current 12%. For instance, the first DCF outcome is found by the following calculation: $-20\,440/1.12 = -18\,250$; the fact that the table contains a slightly different number is that in the tabulated calculations we rounded the numbers only in the end. Finally when we total the DCF-column we find the net present value as Formula (1) prescribed. The idea of a positive NPV is that once NPV > 0 the investment is at least not loss making compared to a 12% investment alternative, under the assumption that in both cases failure is excluded. So if the NPV is positive and large compared to the investment it is a strong indicator to engage in such an investment. Of course when there are a number of such investments to be made, it is possible to rank them by NPV: the highest NPV-ranked investment obtains then the highest priority. In fact, we need to calculate all net present values of all scenarios of all the projects in our portfolio to be able to appraise the IT-portfolio in its entirety.

Scope creep. Table 6 shows the NPV-calculation for the case that there is requirements creep of 2% per month. The development time increases from 6 months to 6.3 months. Also the yearly operational costs rise, since the size of the system to be maintained has been increased. The benefits stay the same, only since we are a little bit later, in the first year these benefits will be lower, hence the 47 500 in the first year. Note that we assumed a proportional decrease in benefits and yearly operational cost if the system is delivered later than planned originally. Since the system becomes available 0.3 months later, $2500 is lost. The higher development costs, higher operational costs, and longer schedule lead to a significantly lower net present value: $76 469 as opposed to $92 257, which is about 17% lower.

Creep and strict deadlines. Of course, when less time means less benefits, decision makers are tempted to keep deadlines strict—which is not always the best thing to do from an investment perspective. For this quite common scenario we also calculated the net present value. Table 7 shows the NPV-calculation for the case that the project suffers both from requirements creep (2% a month) and time compression. Due to this time compression the system is to be delivered within the originally planned schedule of 6 months. So we have higher development costs, more

Table 7
NPV-calculation of project 1 with requirements creep and time compression, so the project duration is kept on its original 6 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	95 320	15 015	50 000	−60 335	−42 235	−37 716
2	0	30 030	100 000	69 970	48 979	39 036
3	0	30 030	100 000	69 970	48 979	34 873
4	0	30 030	100 000	69 970	48 979	31 151
NPV						67 344

Table 8
The rigid NPV for project 2

t	inv	ops	inc	gr C_t	C_t	DCF
1	780 000	0	0	−780 000	−546 000	−487 578
2	0	156 000	4000 000	3844 000	2690 800	2144 568
3	0	156 000	3000 000	2844 000	1990 800	1417 450
4	0	156 000	1000 000	844 000	590 800	375 749
5	0	156 000	1000 000	844 000	590 800	334 984
6	0	156 000	500 000	344 000	240 800	122 086
7	0	156 000	500 000	344 000	240 800	108 842
NPV						4016 101

benefits since we adhered to the deadline, and more operational costs since the software is larger due to the scope extensions. Taking all these effects into account we end up with the lowest value of the three scenarios: $67 344. This is 27% lower than the rigid scenario, so in this case it seems best to mitigate unnecessary scope creep, and if that is impossible not to engage in a death march project where too much work needs to be done in too short a time frame. Namely, from Table 7 it is evident that the higher development costs due to time compression are not compensated by the higher benefits captured in year 1 since the system is delivered 0.3 months earlier. So, in the case of project 1 time-to-market does not pay, in fact it lost 27% on the rigid alternative.

7.2. Supporting private investors

Now let us look at our next project in the sample portfolio in more detail. It concerns the development of a system which enables launching a new, highly competitive financial product on the market. For example, a new investment fund for the private investor. The market analysts estimate the yearly earnings starting after the system has become operational at four million dollars in year 2. The product lifetime is estimated as 6 years. Note that the estimated operational lifetime of a 585-function point system is about 6 years according to benchmark Formula (6). As before we will calculate the rigid, the scope creep, and the time compressed with extended scope scenarios.

Rigid. Table 8 provides an NPV-calculation for project 2 in case there is no volatility in functionality, budget, and duration. In that case the estimated development duration is 12 months according to the benchmark Formula (2). These calculations are completely analogous to the ones in project 1, only with different numbers. As can be seen, the first year of operation a large benefit of four million dollars is projected, which will slowly fade: three million in year 3, then two years one million which decreases further to half a million in the last two years of operation. Still after seven years a large positive net present value is projected of about four million.

Scope creep. Table 9 shows the NPV-calculation for the case that there is requirements creep of 2% a month. The development time increases from 12 to 13.2 months. Also the yearly operational costs rise, since the size of the system to be maintained has been increased. The market analysts estimate a huge loss of earnings in year 2 and year 3 if the system is delivered too late. Since the system is highly competitive, market share is lost in year 2 and year 3 if the system is delivered later than at the end of year 1. In year 2 the loss is $1.5 million and in year 3 it is $0.5 million. This is reflected in the inc-column where different earnings are projected than in Table 8. Indeed the requirements creep extends launching the product, which is harmful since the net present value of this alternative is $2649 998 as opposed to $4016 101 for the rigid scenario, a drop of 34% in value.

Table 9
NPV-calculation of project 2 with 2% requirements creep but no time compression, so with a project duration of 13.2 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	976 782	0	0	−976 782	−683 747	−610 586
2	108 531	178 061	2500 000	2213 408	1549 386	1234 860
3	0	197 846	2500 000	2302 154	1611 508	114 7394
4	0	197 846	1000 000	802 154	561 508	357 119
5	0	197 846	1000 000	802 154	561 508	318 375
6	0	197 846	500 000	302 154	211 508	107 234
7	0	197 846	500 000	302 154	211 508	95 602
NPV						2649 998

Table 10
NPV-calculation of project 2 with requirements creep and time compression, so the project duration is kept on its original duration of 12 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	1532 347	0	0	−1532 347	−1072 643	−957 870
2	0	197 846	4000 000	3802 154	2661 508	2121 222
3	0	197 846	3000 000	2802 154	1961 508	1396 593
4	0	197 846	1000 000	802 154	561 508	357 119
5	0	197 846	1000 000	802 154	561 508	318 375
6	0	197 846	500 000	302 154	211 508	107 234
7	0	197 846	500 000	302 154	211 508	95 602
NPV						3438 275

Creep and strict deadlines. Table 10 shows the NPV-calculation for the case that the project has both requirements creep and time compression. From Table 10 it is evident that the higher development costs due to time compression are fully compensated by the higher benefits captured in year 1 because the system is delivered 1.2 months earlier. So, in the case of project 2 time-to-market pays off, if there is requirements creep as well. Namely, the net present value in Table 10 is $3438 275, which is much higher than the NPV of the requirements creep scenario depicted in Table 9, which amounts to $2649 998. Still the rigid scenario with an NPV of $4016 101 is by far the most desirable. For decision makers it is insightful to know that when scope alterations are unavoidable, *managed* time compression is a way to reduce the loss.

7.3. A new mortgage product

Project 3 also concerns the development of a system enabling the launch of a new financial product on the market. For example, a new mortgage product to be sold on the retail market. The product lifetime is estimated at 6 years. The estimated operational lifetime of a 1036-function point system is about 5.7 years according to the benchmark Formula (6), which is in line with the business projections. The market analysts estimate the yearly earnings starting after the system has become operational at $3 million a year in the first 2 years. The projected yearly earnings drop to $2 million in year 4 and 5 and to $1 million in years 6 and 7. As before, we calculate the various NPVs for the three scenarios.

Rigid. We start as usual with the rigid scenario: Table 11 shows the NPV-calculation for the zero-tolerance case: no changes to functionality, time, and budget. The estimated development duration is 15 months in that case. This is reflected in the benefits and operational costs as well: since we have 3 months non-operational time in year 2 the income will be proportionally lower in this case: $2.25 million, which is 3/4 of the projected $3 million. We end up with a positive value of $3378 451.

Scope creep. Table 12 displays the NPV-calculation for the case that there is requirements creep of 2% a month. Consequently, the development time increases from 15 months to 16.8 months. Also the yearly operational costs rise, since the size of the system to be maintained becomes larger. However, the earnings in the first 2 years after delivery of the system are projected to surge, because the quality of the mortgage product is improved to a large extent by

Table 11
The rigid NPV for project 3

t	inv	ops	inc	gr C_t	C_t	DCF
1	1381 334	0	0	−1381 334	−966 934	−863 472
2	345 333	207 200	2250 000	1697 467	1188 227	947 017
3	0	276 267	3000 000	2723 733	1906 613	1357 509
4	0	276 267	2000 000	1723 733	1206 613	767 406
5	0	276 267	2000 000	1723 733	1206 613	684 150
6	0	276 267	1000 000	723 733	506 613	256 853
7	0	276 267	1000 000	723 733	506 613	228 989
NPV						3378 451

Table 12
NPV-calculation of project 3 with requirements creep but no time compression, so with a project duration of 16.8 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	1858 666	0	0	−1858 666	−1301 066	−1161 852
2	743 467	223 040	2400 000	1433 493	1003 445	799 746
3	0	371 733	4000 000	3628 267	2539 787	1808 328
4	0	371 733	3000 000	2628 267	1839 787	1170 105
5	0	371 733	3000 000	2628 267	1839 787	1043 159
6	0	371 733	1500 000	1128 267	789 787	400 422
7	0	371 733	1500 000	1128 267	789 787	356 984
NPV						4416 891

Table 13
NPV-calculation of project 3 with requirements creep and time compression, so the project duration is kept on its original duration of 15 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	3173 655	0	0	−3173 655	−2221 559	−1983 852
2	793 414	278 800	3000 000	1927 786	1349 450	1075 512
3	0	371 733	4000 000	3628 267	2539 787	1808 328
4	0	371 733	3000 000	2628 267	1839 787	1170 105
5	0	371 733	3000 000	2628 267	1839 787	1043 159
6	0	371 733	1500 000	1128 267	789 787	400 422
7	0	371 733	1500 000	1128 267	789 787	356 984
NPV						3870565

processing the requests for requirements change and addition submitted during the project execution. This business improvement will trickle down to earnings in subsequent years as well. In fact, the market analysts project earnings of four million per year in the first two operational years. Then two years with projected earnings of $3 mn, and the last two years $1.5 mn is projected for this product.

This is reflected in the inc-column of Table 12. Indeed as can be seen, the requirements volatility that will cost an extra investment does pay off, since a higher net present value is obtained in the end: $4416 891 which is higher than the rigid scenario, which amounts to $3378 451, so extension of the requirements increase the net value with a million dollar. Investing more incurs more costs, but this leads to an even higher NPV. Hence, the popular cost-leadership strategy that some decision makers apply by default to information technology is harmful in this scenario. This becomes apparent when you look at a larger value chain than only a cost target for the IT-department.

Creep and strict deadlines. Table 13 depicts the NPV-calculation for the case that the project suffers both from requirements creep and is built under time compression, since we stick to the original deadline of 15 months. Clearly, time compression does not pay off: the net present value dropped to $3870 565, from $4416 891, which would cost us about half a million dollar. So speed-to-market is in this case not a winning strategy, despite the fact that more earnings are due within the first two years.

Table 14
The rigid NPV for project 4, which takes two years to implement

t	inv	ops	inc	gr C_t	C_t	DCF
1	4611 000	0	0	−4611 000	−3227 700	−2 882 336
2	4611 000	0	0	−4611 000	−3227 700	−2 572 477
3	0	922 000	10 000 000	9078 000	6355 460	4 524 475
4	0	922 000	9 000 000	8078 000	5654 600	3 596 326
5	0	922 000	9 000 000	8078 000	5654 600	3 206 158
6	0	922 000	9 000 000	8078 000	5654 600	2 866 882
7	0	922 000	9 000 000	8078 000	5654 600	2 555 879
8	0	922 000	9 000 000	8078 000	5654 600	2 284 458
9	0	922 000	7 000 000	6078 000	4254 600	1 535 911
10	0	92 2000	5 000 000	4078 000	2854 600	919 181
NPV						21 489 270

Creep can pay off. From Table 12 it is evident that the higher development costs in addition to the higher yearly cost of operation due to requirements creep are fully compensated by the higher benefits captured in year 2 and 3 by selling a higher quality mortgage product. So, in this example requirements creep is not a burden but an advantage. The net value of the requirements creep scenario is the highest. Indeed, the rigid scenario where no alterations to the requirements are allowed has lower development and operational costs, but is not as beneficial for the business as a change-of-plan which is reflected in Table 12. In the case of offshore development, where zero-change governance rules are sometimes used to prevent uncontrolled cost-surges, it is not always a prudent strategy to engage in development efforts where requirements volatility can be beneficial. For, those added benefits are curtailed by the zero-change policy. Then again, if you foresee an IT-investment with potentially high volatility and higher cash-flows, it is maybe not the best strategy to offshore its development at all.

7.4. A new payment system

Project 4 concerns the development of a very large system, for example a new payments system. After careful market research, the market analysts estimate the yearly earnings starting after the system has become operational at ten million dollars in the first year, and the subsequent 5 years at nine million dollars a year. Then earnings drop somewhat to seven million, and in the last year to $5 mn. The market analysts estimate product lifetime as 8 years. We note that the estimated operational lifetime of a 3460-function point system is also about 8 years according to benchmark Formula (6). We will provide again three tables with various cash-flows that lead to various net present values.

Rigid. Table 14 summarizes the rigid scenario estimate of the net value given the cash-flows predicted by the business. The NPV is about 21.5 million dollars which is high, but the investment is also considerable: over nine million dollars spread over two years. Note that we assumed that the system is good when it is delivered, and that no additional maintenance costs are necessary for improvements and enhancements. This is not always realistic, but since we left out this factor in all cases, we can still compare alternatives.

Scope creep. Table 15 contains the full NPV-calculation for the case that there is requirements creep of 2% a month. Consequently, development time increases from two years to 28.9 months. Also the yearly operational costs rise, since the size of the system to be maintained increases. The benefits that the market researchers projected do not change by the added requirements, so these may be useful to some of the stake holders, but they do not give rise to better earnings. This is strongly reflected in the value, since this scenario is appraised at about $8 mn which is more than twice as less as the rigid scenario.

Creep and strict deadlines. Usually executives tend to stick to the original plan with respect to deadlines—even in the case of changing the scope of a project. This is reflected in the appraisal of this scenario of our new payment system: it is going to be lossy in this scenario. Table 16 provides evidence of this unwanted situation. The NPV-calculation illustrates that when project 4 suffers both from requirements creep and is built under time compression, the net value

Table 15
NPV-calculation of project 4 with requirements creep but no time compression, so with a project duration of 28.9 months

t	inv	ops	inc	gr C_t	C_t	DCF
1	7417 516	0	0	−7417 516	−5192 261	−4636 689
2	7417 516	0	0	−7417 516	−5192 261	−4138 232
3	3028 818	878 063	6000 000	2093 119	1465 183	1043 211
4	0	1484 051	9000 000	7515 949	5261 164	3346 100
5	0	1484 051	9000 000	7515 949	5261 164	2983 080
6	0	1484 051	9000 000	7515 949	5261 164	2667 410
7	0	1484 051	9000 000	7515 949	5261 164	2378 046
8	0	1484 051	9000 000	7515 949	5261 164	2125 510
9	0	1484 051	7000 000	5515 949	3861 164	1393 880
10	0	1484 051	5000 000	3515 949	2461 164	792 495
NPV						7954 811

Table 16
NPV-calculation of project 4 with requirements creep and time compression, so the project duration is kept on its original duration of two years

t	inv	ops	inc	gr C_t	C_t	DCF
1	17 806 663	0	0	−17 806 663	−12 464 664	−11 130 944
2	17 806 663	0	0	−17 806 663	−12 464 664	−9 934 337
3	0	1484 051	10 000 000	8 515 949	5 961 164	4 244 349
4	0	1484 051	9 000 000	7 515 949	5 261 164	3 346 100
5	0	1484 051	9 000 000	7 515 949	5 261 164	2 983 080
6	0	1484 051	9 000 000	7 515 949	5 261 164	2 667 410
7	0	1484 051	9 000 000	7 515 949	5 261 164	2 378 046
8	0	1484 051	9 000 000	7 515 949	5 261 164	2 125 510
9	0	1484 051	7 000 000	5 515 949	3 861 164	1 393 880
10	0	1484 051	5 000 000	3 515 949	2 461 164	792 495
NPV						−113 411

is negative, and a loss of 113 411 dollars is projected. Again, we assumed a proportional decrease in benefits and yearly operational cost if the system is delivered later than planned originally. So the projected benefits in year 3 in Table 15 are lower than in Table 14 and Table 16.

A mission-impossible project. From Table 16 it is evident that the higher development costs due to scope creep and time compression are not compensated by the higher benefits captured in year 3 because the system is delivered 4.9 months earlier. So, in the case of project 4 time-to-market does not pay off. In fact, this is a lossy scenario, one that we found all too often, when originally the business case looked very promising, as in this example where a projected NPV of tens of millions is calculated. But when the IT-risks of requirements creep compounded with time compression are not mitigated, the entire project becomes a financial failure.

8. Appraising the portfolio

We calculated the various gains for the sample IT-investment portfolio consisting of four projects of varying size. We did not consider the possibility that implementation of part of this portfolio fails, in other words; one or more of the IT-projects can be challenged or never materialize. But as soon as we take into account the possibility of project failure, the yield is no longer a deterministic variable but a random variable. Instead of just calculating the net present value we now must calculate the (expected) yield, as we discussed in the introduction. The purpose of this section is to show how to calculate the yield of the individual projects and lift that to the portfolio level.

For completeness' sake we summarize the gain matrix in Table 17. It consists of the various outcomes of the appraisals of the individual projects for the 3 scenarios we already discussed. We will number the scenarios as follows.

1. Rigid.
2. Requirements creep at 2% per month but no time compression.
3. Both requirements creep at 2% a month and time compression.

Table 17
The gain matrix in dollars for the sample portfolio for different scenarios

	Project 1	Project 2	Project 3	Project 4
Scenario 1	92 257	4016 101	3378 451	21 489 270
Scenario 2	76 469	2649 998	4416 891	7 954 811
Scenario 3	67 344	3438 275	3870 565	−113 411

Table 18
Loss in dollars of our projects in all scenarios in case of failure

	Project 1	Project 2	Project 3	Project 4
Scenario 1	−41 257	−487 578	−882 738	−5 454 813
Scenario 2	−49 367	−671 135	−1576 632	−10 284 484
Scenario 3	−59 585	−957 870	−2426 498	−21 065 281

Of the three distinguished scenarios 2 and 3 occur frequently in in-house situations, whereas scenario 1 is often seen in outsourcing and offshoring. It is worth noting that scenario 3 is also typical for government. Of course, when the outsourcing relation is not managed properly, they reduce to scenarios 2 and 3, but since there are contracts in between, this does not continue indefinitely. These often occurring scenarios can be used to compare to one another, and to monetize the costs or benefits with respect to alternative scenarios.

8.1. The yield distribution

Now we take project failure into account. For this portfolio we assume that in the case of project failure there are no benefits from the investment. Moreover, we assume that in the failing case the loss consists of the discounted net development costs. Since we used a tax break of 30%, the net development cost equals to 70% of the gross development cost. In Table 18, we enumerated the loss matrix consisting of the various losses. Let us explain the numbers in the first row. For example, the first amount for project 1 in scenario 1 is −$41 257. It is obtained by discounting 70% of the development costs of project 1, which was $66 000 in year 1 (see Table 5). This amounts to $0.893 \cdot 0.70 \cdot 66\,000 = \$41\,257$. The used constant 0.893 stems from Table 25 (Section 10) providing discount factors to use in the NPV-calculations. For projects 2 and 4 the discounted value of the development costs in scenario 1 can be taken directly from Tables 8 and 14, respectively. The reason is that unlike the calculation for the first project, in the other projects no income and operational costs were made and the development costs were made in one year. Thus, we can use the number in the column since it consists purely of the discounted development costs. Project 3 is somewhat more complex. First we take the pure net development costs from Table 11. But in the next year we have more development costs, and we discount them as in the case of project 1. In the second year we invest another $345 333, with the tax break of 30% and the discount factor for the second year 0.797 (see Table 25) we obtain another net $19 266, which totals to a loss of $882 738. In the same way we calculated all the other elements of the loss matrix in Table 18: for each scenario we discounted the development costs. As can be seen, some of the losses can be quite large, and the maximal loss in this table is over $21 mn. This is a large amount and given a worldwide annual loss of $290 bn on failing IT-projects [9] on an estimated total $1.16 trillion IT-spent (according to a 2007 IDC report), such risks evidently materialize, and therefore must be accounted for in appraisals of IT-portfolios.

If a project succeeds the NPV of the project is gained. Let L stand for the loss in the case of project failure. Let cf be the chance of failure of a project p. Then the yield Y_p of project p is the following two-point or Bernoulli distribution:

$$Y_p = \begin{cases} L, & \text{with probability } cf \\ \text{NPV}, & \text{with probability } (1 - cf). \end{cases} \tag{10}$$

The expected yield $E(Y_p)$ of this project can be calculated as follows:

$$E(Y_p) = cf \cdot L + (1 - cf) \cdot \text{NPV}.$$

In words: the expected yield of a project is the loss times the chance it occurs plus the gain in case of success. Lifting this to the portfolio level is not hard. The yield $Y(P)$ of an IT-portfolio P with n IT-projects is defined as the

Table 19
The expected yields for the four projects in all scenarios

	Project 1	Project 2	Project 3	Project 4
Scenario 1	85 501	3429 722	2646 805	13 818 290
Scenario 2	69 624	2163 784	3237 965	1864 710
Scenario 3	60 439	2794 679	2631933	−7 109 240

following (discrete) 2^n-point distribution:

$$Y(P) = \sum_{p \in P} Y_p.$$

The expected yield of an IT-portfolio P is simply the expected value of $Y(P)$:

$$E(Y(P)) = E\left(\sum_{p \in P} Y_p\right) = \sum_{p \in P} E(Y_p).$$

So the expected yield of the portfolio is just the sum of the expected yields of the individual projects. For our IT-projects we have a function calculating the chance of failure: it is Formula (7). In other cases, other formulas are perhaps more appropriate, and in case of internal data, organization-specific benchmarks can be used. Whatever means you use to estimate the chance of failure, calculating the expected yield is as in the above equations.

As an example we calculate the expected yield $E(Y_2)$ of project 2 if it is developed under scenario 3. Then we find:

$$E(Y_2) = 0.1464 \cdot -957\,870 + 0.8536 \cdot 3438\,275 = \$2794\,679.$$

Here the chance of failure is taken from Table 4 where the larger size is taken into account since there is requirements creep: 0.1464. The loss is taken from Table 18. The success rate is simply found by inverting the failure chance of 0.1464 times the calculated gain for project 2 in scenario 3, which we summarized in Table 17.

Table 19 shows that the expected yields are much lower than the deterministic gains in terms of their NPVs for all projects and all scenarios. The difference can be interpreted as the quantified loss exposure of the projects and provides useful information to the IT-governors to decide upon investing or not investing in the development of the IT-portfolio. One can also assume that the loss in case of project failure is 50% of the development cost and not 100%, in case the failure already became apparent halfway the development cost spending. In that case the expected yield will be higher. But unless there is effective kill management, IT-failures are characterized by large cost and time over-runs *before* cancelling the project, so our approach to use for the loss 100% of the planned development budget is conservative.

8.2. From project to portfolio level

We now make the step from individual projects to the portfolio level. We explain how to do this via our running example of the small portfolio consisting of the four IT-projects. Let us assume the rigid scenario for the entire portfolio. So there is no change in functionality, cost, and schedule. If the chance of failure of the projects is left out of consideration the yield of the entire sample portfolio simply equals the sum of the gains (being the deterministic NPVs) of the individual projects. If we take into account the chance of failure of the projects, the portfolio yield becomes a random variable Y. We will now show how one can calculate the probability distribution of this random variable. For the sake of simplicity, we assume that the failure rates of the projects are independent from each other. For our sample portfolio there are $2^4 = 16$ different possible outcomes of our yield distribution. For each possible portfolio yield we can calculate the corresponding probability of its occurrence. The yield at which the gain is at its maximum is $92\,257 + 4016\,101 + 3378\,451 + 21\,489\,270 = \$28.98$ million and is obtained when all projects succeed. This is just the sum of the elements of the first row of Table 17. The probability connected with this occurrence is:

$$0.9494 \cdot 0.8698 \cdot 0.8283 \cdot 0.7153 = 0.48927.$$

The numbers are derived from Table 4 as follows: in the third column the chance of failure without requirements creep is given. So the chance of success is the inverse, e.g., $1 - 0.0506 = 0.9494$ is the chance of success for

Table 20
The yield Y, its probability distribution function, and its cumulative distribution function for our small sample portfolio (the yield is in millions of dollars)

Y	PDF	CDF
−6.87	0.00032	0.00032
−6.73	0.00604	0.00636
−2.61	0.00155	0.00792
−2.47	0.02915	0.03707
−2.36	0.00215	0.03922
−2.23	0.04037	0.07959
1.90	0.01038	0.08997
2.03	0.19473	0.28470
20.08	0.00081	0.28551
20.21	0.01518	0.30069
24.34	0.00390	0.30459
24.47	0.07324	0.37783
24.58	0.00541	0.38324
24.72	0.10142	0.48466
28.84	0.02608	0.51073
28.98	0.48927	1.00000

the first project, and so on. Since the projects were assumed to be independent we can multiply the various success probabilities, and end up with little under 50% chance that the entire portfolio materializes without any project failure. Is that very low? No, since longitudinal benchmark data from Standish group say that on average 20% of the projects are successful, 50% are challenged, and 30% are cancelled [16–18]. Since half of the projects in our sample portfolio are relatively small, success rates are higher, hence the relatively high outcome of over 50%.

The lowest possible portfolio yield occurs when all the projects fail. Then the loss equals the sum of the discounted development costs after tax deduction of 30% of the four projects. Just take the sum of the scenario 1 row in Table 18, which amounts to: $-41\,257 - 487\,578 - 882\,738 - 5454\,813 = -\$6.87$ mn.

In an analogous way the probabilities connected with all possible 16 portfolio yields can be calculated. For example, the probability of a yield of $92\,257 + 4016\,101 - 882\,738 - 5454\,813 = -\$2.23$ million, which corresponds with the occurrence: project 1 and project 2 succeed and project 3 and project 4 fail (all under scenario 1), equals to: $0.9494 \cdot 0.8698 \cdot 0.1717 \cdot 0.2847 = 0.0404$. The complete probability distribution of the portfolio yield of our running sample portfolio is listed in Table 20.

Fig. 4 visualizes the discrete probability function of the portfolio yield: it simply visualizes the table entries. The density in Fig. 5 tells us something about the nature of the probability function that is potentially underlying the phenomenon in question. In this case the shape of the non-zero yield densities are more or less 4 spots representing the four major non-zero concentrations in the discrete case. Its interpretation is as follows: to calculate the *probability* of certain outcomes we have to calculate the integral under the surface of the density function. So there is no direct interpretation in terms of probabilities for density functions. But it is clear that we are not dealing with a density that represents some of the known theoretical shapes, or their superposition.

Cumulative distribution function. From column 2 of Table 20 we can easily derive the cumulative distribution function of the portfolio yield, giving the probability that the outcome of the portfolio yield is a value not exceeding a certain value. In mathematical terms, for a discrete density function $f(x_i)$, with $i = 1, 2, \ldots$, the cumulative distribution function $F(x)$ is defined by:

$$F(x) = \sum f(x_i),$$

where the summation is over those values of i such that $x_i \leq x$. In Fig. 6 we depict the graph of the cumulative distribution function F for our small sample IT-portfolio. The cumulative distribution function gives us all kinds of information regarding the probabilities associated with various outcomes of the portfolio yield. For example, the first quartile is found from the cumulative distribution function by calculating for which of the possible values of x does $F(x)$ equal or exceed 0.25. A portfolio yield of \$1.90 million gives us the desired quartile. This implies that in

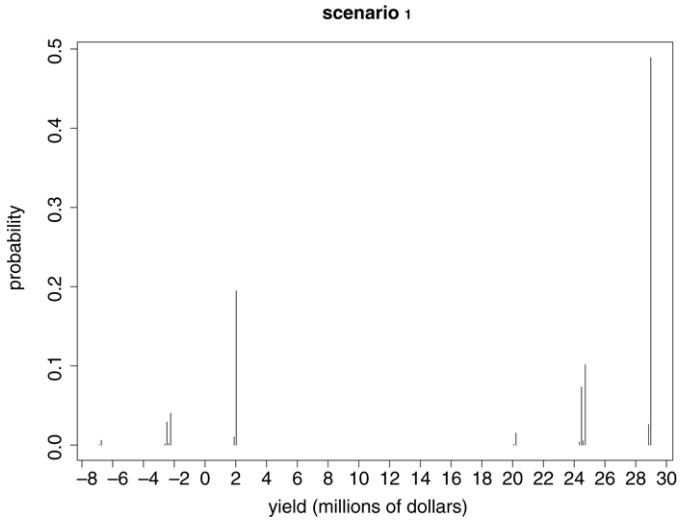

Fig. 4. Discrete density function of the yield of our sample portfolio for the rigid scenario without time compression and requirements creep.

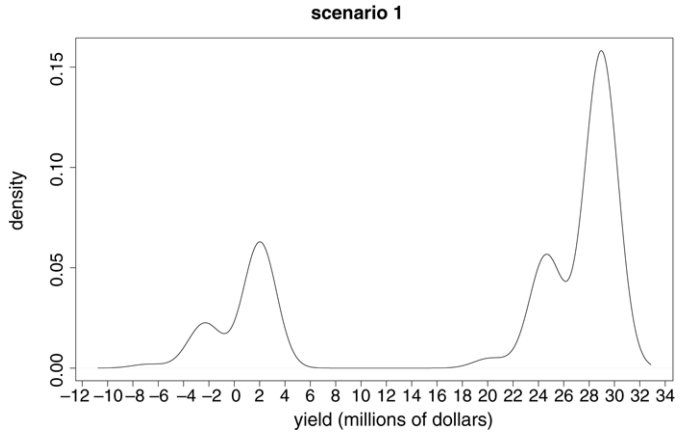

Fig. 5. Probability density function of yield in the rigid scenario.

25% of the possible outcomes the portfolio yield can be below two million, roughly. Analogously, we can calculate other probabilities, depending on the interests of the decision makers.

Box plot. Apart from the empirical plots expressing the discrete probability, the probability density function, and the cumulative distribution, it is also insightful to summarize the data both visually and in tabular format. We do this via a so-called box and whiskers plot, or box plot [32,27]. A box plot is just a visual form to summarize the data with as few points as possible. One well-known summarizing value is the median, dividing the data into two equally-sized groups. With the so-called *quartiles* we divide data into four equally-sized groups providing a fairly good idea of the distributional properties [32,27]. In Fig. 7, we depict for all scenarios of our four-project sample portfolio their accompanying box plot. The rectangular boxes represent 50% of the data, so the bulk. The thick horizontal lines are the medians.

What strikes in the box plot in case of the first scenario is that the median is just below the upper side of the box. In fact this means that the median value of $28 842 565 is almost the maximal value, being $28 976 079. So there is a 50% chance that the portfolio yield of this small IT-portfolio is between 28 842 565 and 28 976 079 dollars. Another insight is that the chance of having an investment with a positive yield is very large: the box plot plus upper part represents 75%, together with the part from the bottom of the box to the zero-yield-line we end up with a chance of 92% that the yield is going to be above zero. So, there is a very large chance that this portfolio is going to satisfy the criterion of a positive yield, meaning that it is a good investment, with a relatively small chance of being lossy.

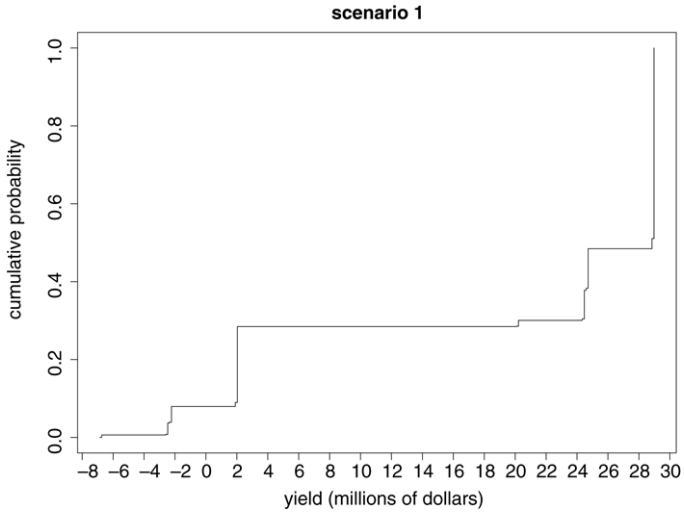

Fig. 6. The cumulative distribution function F for our small four-project IT-portfolio, assuming the rigid scenario.

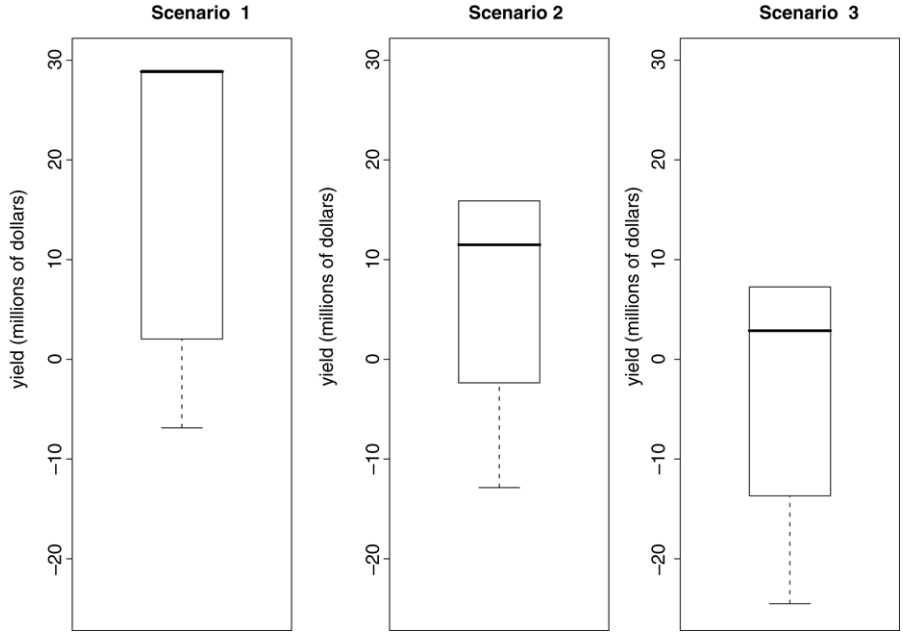

Fig. 7. Three box plot for the four-project sample portfolio, representing summaries of the yield distributions for all three scenarios.

You cannot always use a box plot alone to conclude on this, as we will see shortly, but if the entire box is above the zero-yield-line it is fair to say that the investment scenario is sound.

All box plots use the same vertical axis to make comparisons between the scenarios simpler. The second box plot in Fig. 7 is much lower than the first, and the median is no longer almost the maximal value. Also the box itself is within the zero-line meaning that we have to be more careful with the fact that this box plot suggests a chance of 66.6% that the yield ≥ 0. This is because a box plot does not illustrate where the non-zero densities are concentrated. We use the probability density functions to assess that. Scenario 3 is represented by the lowest box plot in Fig. 7: the box representing 50% of the data points is for the most part under the zero-yield-line, meaning that using this scenario, the investment is maybe not the best idea. Also in this case the chance that the yield is above zero is substantial: 65%, so almost as high as scenario 2. Also here we have to be careful with this finding since the PDF of the investment portfolio is highly heterogeneous. Nevertheless, scenario 1 is by far the best for this investment.

Table 21
Summary statistics for the three scenarios of our four-project example portfolio

#	Min	1st qu	Med	Mean	3rd qu	Max
1	−6 866 386	2 031 996	28 842 565	19 980 318	28 976 079	28 976 079
2	−12 868 353	−2 352 849	11 490 301	7 966 978	15 886 446	15 886 446
3	−24 509 234	−13 689 097	2 866 628	−1 622 189	7 262 773	7 262 773

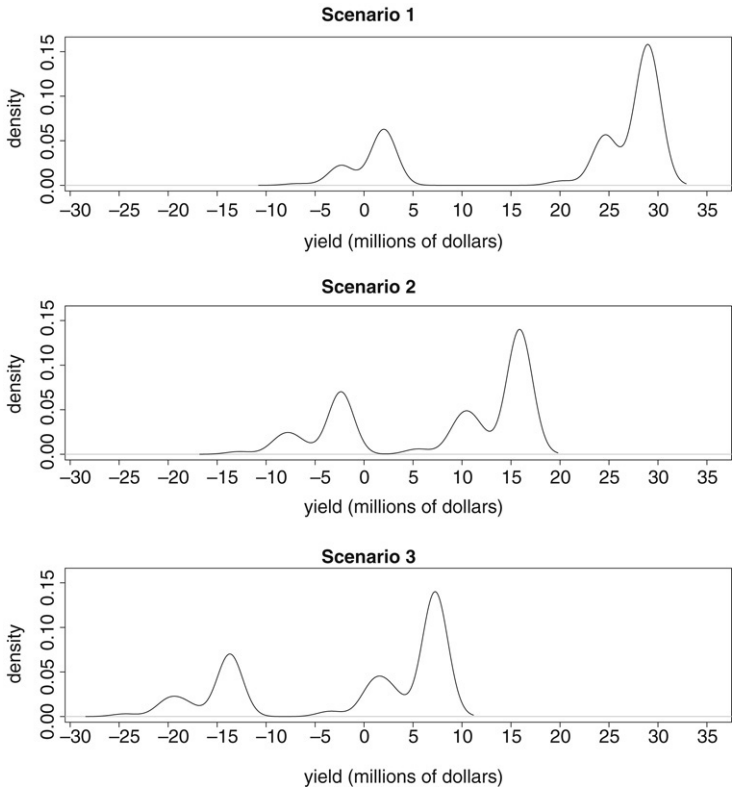

Fig. 8. Three PDFs for the four-project sample portfolio.

In Table 21 we summarized the numerical values that we depicted via the box plot in Fig. 7. As a rule of thumb, you should be careful to draw conclusions based on the summaries, be it visual or numerical, if the median and the mean are wildly differing. This is the case for all three scenarios as can be seen from Table 21. Therefore, we will also examine the three PDFs belonging to the three scenarios in more detail. We plot them in Fig. 8. We aligned the horizontal axes so that it is easy to see where the non-zero density of the PDFs is located. Indeed, if we look at the zero-line, we observe for the first plot in Fig. 8, representing scenario 1, that there is a non-zero density at zero and beyond, leading to a large probability that the portfolio yield of this investment is going to be positive under scenario 1. If we look at scenarios 2 and 3, we note that they are roughly translated along the horizontal axis to the left. The specific shape of the density is that in scenario 2 a large part of non-zero density starts, where as for scenario 3 this part ends. That is the reason why for the box plots we do not see any difference in the chance of having a positive portfolio yield. Still, we clearly observe from the PDFs of both scenarios that they are less favorable, and that the chance of having a low to negative portfolio yield is much larger than that in scenario 1.

Recall that for densities the probabilities are given by the areas under the curve, not by the values of the function. Therefore, the density is less practical in calculating probabilities from. We approximate the probabilities via a histogram. In Fig. 9, we depict for each scenario these histograms. A histogram can be read as follows: the probability that the yield of the four-project portfolio in a rigid scenario is between 28 and 30 million dollars which is about 50%. We know this by inspecting the most-right bar with a height of about 0.5, or 50%. Notice that this is more or less the

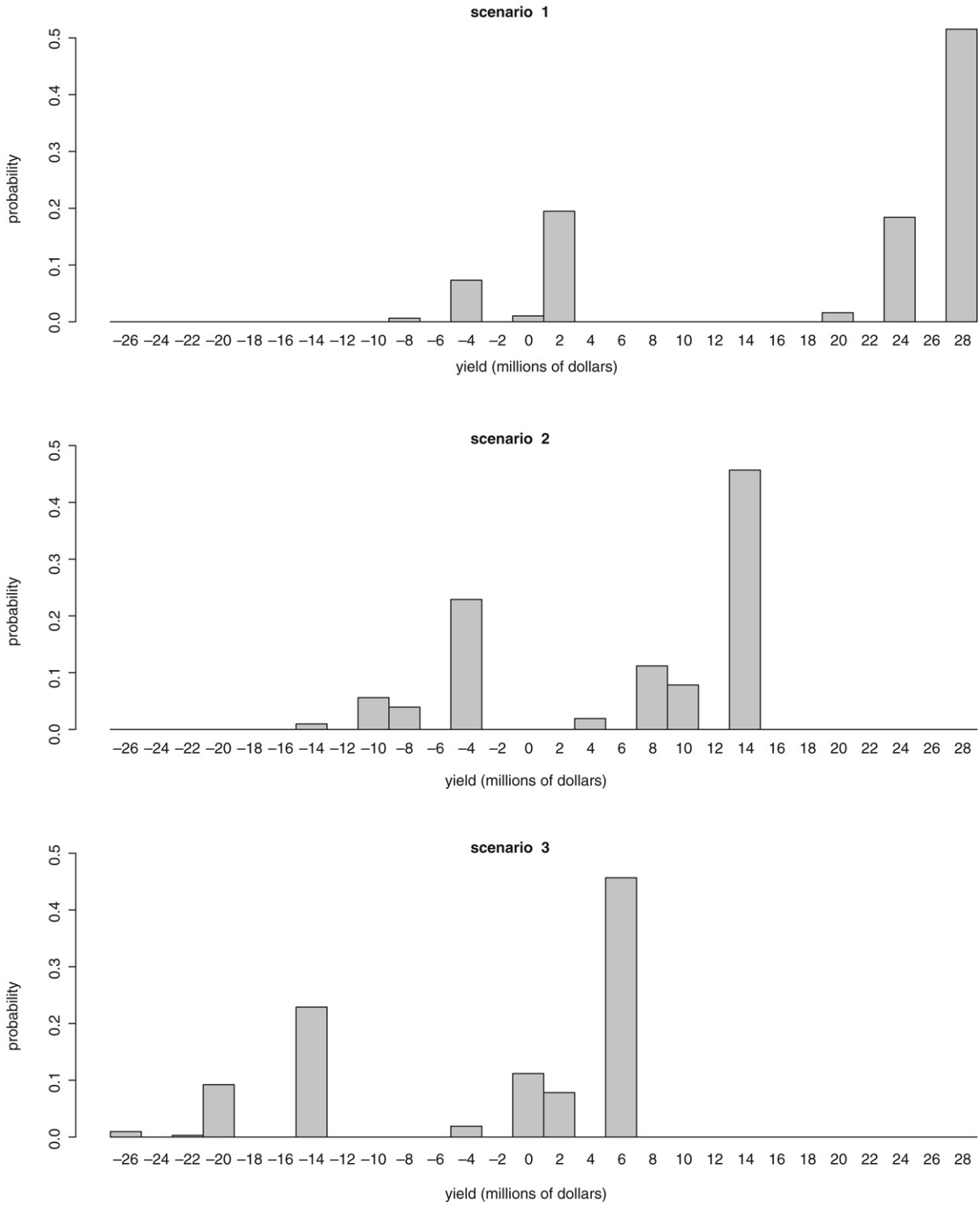

Fig. 9. Three histograms for the four-project sample portfolio.

same as in Fig. 5, where the discrete distribution is depicted. The same chance for scenario 2 is zero, since there is no bar at that interval. And from plot 3 we conclude that although the chances are substantial that there is a positive yield, the chance that this yield is going to be near 28 million is zero, but will be at 7 to 8 million dollars with a 50% chance.

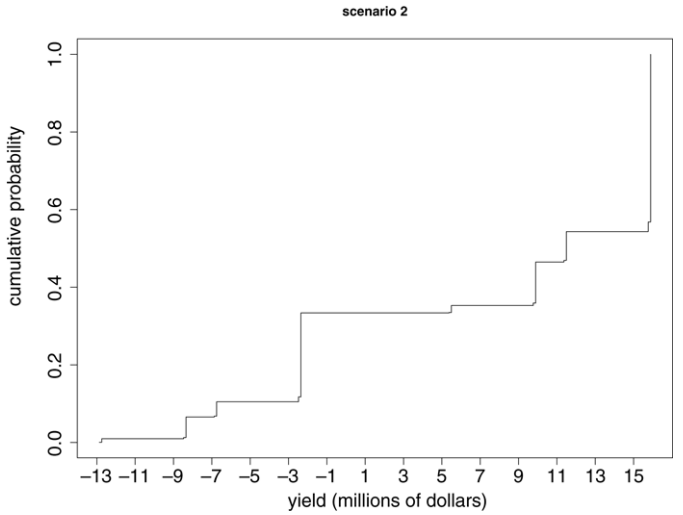

Fig. 10. The cumulative distribution function F for our small four-project IT-portfolio, assuming a requirements creep scenario, but no uniform time compression.

8.3. Other scenarios

We explained in detail how to calculate the yield for the various scenarios, and we looked at comparisons between them. In the same way the discrete, continuous, cumulative distributions, box plots, and histograms of the portfolio yield of IT-portfolios can be calculated and visualized for other scenarios. For example, we provide in Fig. 10 the cumulative distribution function in case of scenario 2 in which there is requirements creep of 2% per month. To calculate the expected yield we first take the highest gain for each project. This implies for project 2 that we should also use time compression since that projects a higher gain. In Fig. 10 we depict the cumulative distribution function of the yield of our sample portfolio in case there is requirements creep in the portfolio. Note that often entire portfolios are subjected to requirements creep. It is our experience that if there is no formal change control in place, there will be requirements creep irrespective of added value or value destruction. When requirements creep is consciously managed, you fall back to a rigid scenario, where in some cases a higher gain for a single project is to have some managed time compression or controlled requirements volatility. In this case, there is apparently requirements creep, and we consciously manage time compression by allowing it for project 2 but not for the other three, since this will diminish the yield per project, leading to a lower portfolio yield than possible.

9. Valuing a realistic portfolio

In principle, we explained by way of our small running example consisting of four projects of different size, with different cash-flows for different IT-uncertainty scenarios how to characterize the stochastic nature of its yield. Depending on the decision criteria the highest yield can be projected. In this section we illustrate the results of our calculations on a larger example. We will not be able to tabulate all intermediate results, since the number of possibilities explodes when more IT-projects enter the portfolio. In this case we take an IT-investment portfolio of 20 projects, with a total investment cost of 37 million dollars. The number of projects leads to maximally $2^{20} = 1048\,576$ possible outcomes for the yield distribution per scenario. From that we derive for each scenario the important pictures as before: the discrete distributions, the densities, the histograms, the box plots, and the cumulative distributions. This provides us with the tools to compare the three scenarios. The three scenarios can be seen as extremes: either entirely rigid, requirements creep all over, or volatile requirements in the presence of strict deadlines. Those three scenarios do occur in practice, but also a continuum of situations in between. Depending on which factual scenario approximates one of the extremes best, you can interpolate our visualizations for intermediate situations. When a lot of money is at stake, or an intermediate scenario flips from a positive to a negative yield, it is necessary to calculate that special scenario in detail. The methods we propose can be used for that without any alteration.

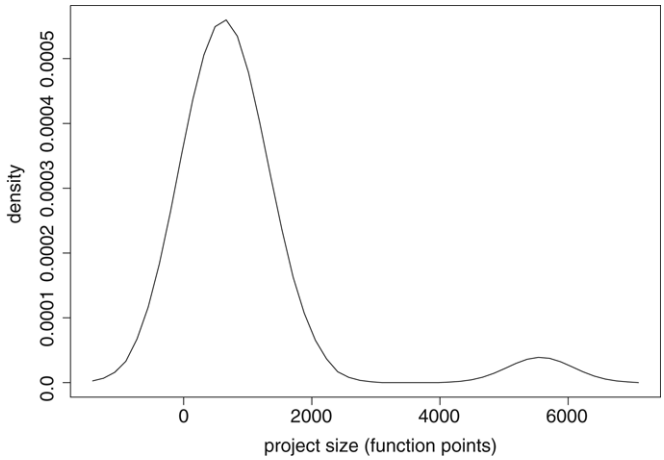

Fig. 11. Density function of our 20-project portfolio resembling a Generalized Pareto Distribution.

The 20-project portfolio is composed of the following types of projects.

- Six-projects resembling our example project 1. Small systems in the 100 function point range, with cash-flows similar to what we have seen. Moreover risky scenarios with scope extension and time compression are not paying off.
- Ten-projects similar to our example project 2. Systems in the 500 function point range, where requirements creep often does not lead to more benefits. However, in case there is scope volatility, time compression is a way to reduce loss since our projects are time-critical because they concern new business.
- Three-projects characterized by our sample project 3. They concern medium-sized systems in the 1000 function point range. They all have somewhat longer development schedules of at least a year, and they can and often do benefit from some healthy requirements change. Namely, the environment will change within a year so inevitably the software needs to evolve, too, even if it is is not operational yet. However, they mostly do not benefit from reducing the development schedule to the originally planned time frame. Usually, the competitive edge is not in a quick win, or capture of a market share to outplay the competitor but those systems rely on larger transaction volumes and their quality. Higher quality leads to less exceptions to be done by hand, and thus more yield.
- A single project of type 4. In our running example this was a new payment system, but it can also be a new brokerage operation, an insurance claims system, and other large long-lived systems. They are large IT-investments in their own right, and their size is in the 5000 function point range. Development schedules are long, the risk of failure is high. For such large projects meticulous planning, requirements engineering, formal change control, and more are necessary. Such large projects need proper risk mitigation, and requirements creep and/or time compression are failure factors, rather than manageable risks that can boost the yield. Also in our running example this became apparent. The rigid case of project 4 had a healthy positive net present value, but the compounded risk scenario where time compression amplifies the problems due to requirements creep projected a negative NPV.

Industrial IT-portfolios contain often a lot of small projects, but also a heavy tail with large ones. This is also reflected in the 20-project portfolio. In Fig. 11 we show the density plot of the project sizes. The shape of the distribution is characteristic for other real-life IT-portfolios. In paper [34], we provide evidence that a lot of important software KPIs have a distribution that is asymmetric, leptokurtic, and possibly heterogeneous with a heavy tail. Looking at Fig. 11, we note that its shape is indeed asymmetric, and the spike is smaller than the bell curve (which is leptokurtic), there is some heterogeneity, and a long and heavy tail.

Also for this portfolio we assumed that the failure rates of the projects are independent from each other. The independence assumption may not be reasonable if too many projects have to be carried out by a relatively small IT-staff, especially when more larger projects are carried out by the same staff members. Then the portfolio size density compared to densities of low-risk portfolios will reveal the problems, and hence an investment calculation is not the best way to proceed. Instead, a risk mitigating strategy is necessary to untangle the heavy correlations so that risk is better diversified. It often helps to break down larger projects into a series of smaller ones, and if that is not possible for some of the larger projects, more experienced staff and management is necessary to manage these risky endeavors.

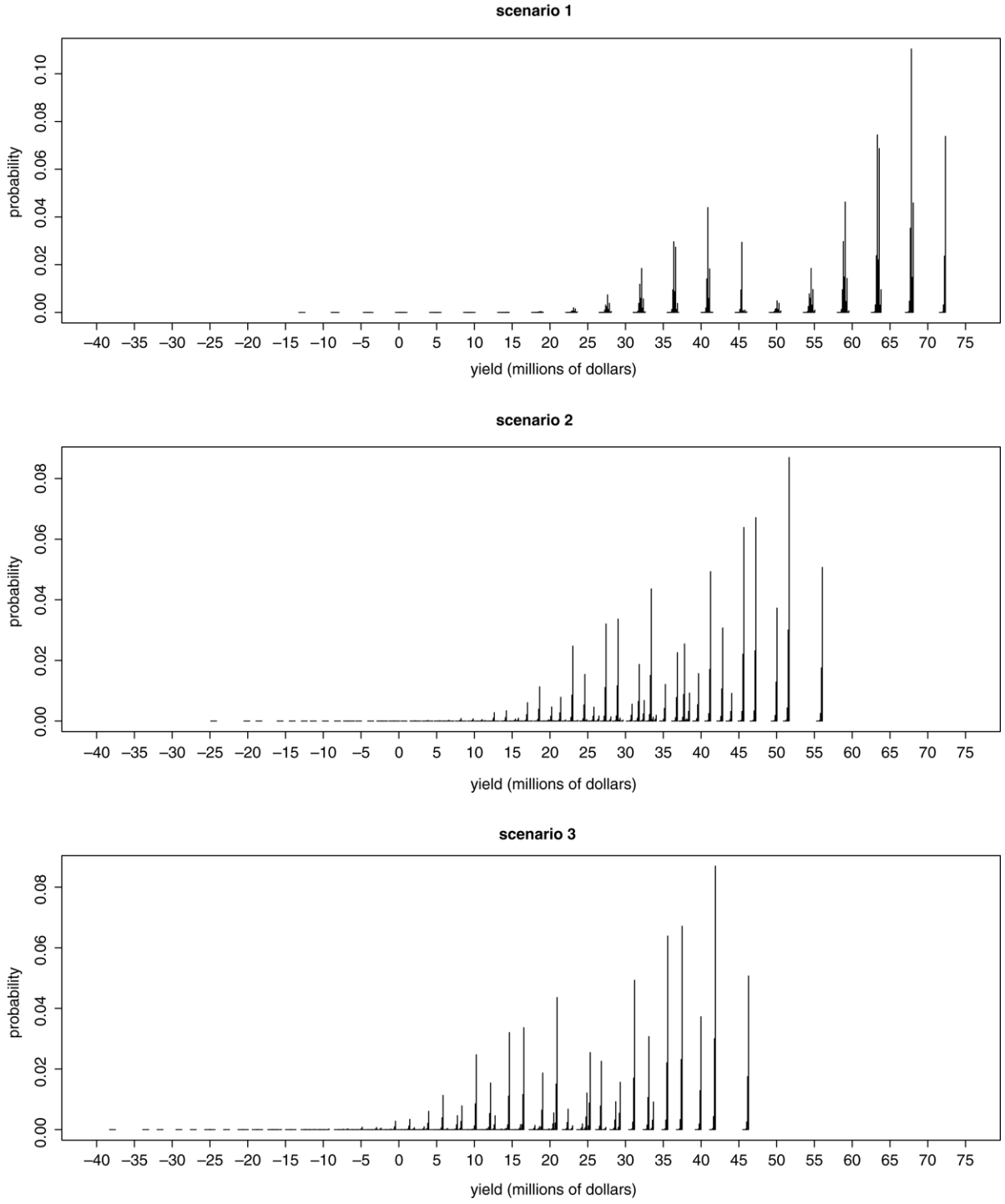

Fig. 12. Discrete density distributions of the portfolio yield of a 20-project portfolio for all three scenarios.

In Fig. 12, we depict the discrete density distributions of the portfolio yield for all the three scenarios. All plots show a long tail to the left indicating a relatively large non-zero chance that the IT-investment portfolio can turn out to be lossy. But since most of the non-zero density is above zero, there is good hope for this investment portfolio. In our 4-project example we had only a few possibilities, leading to a very heterogeneous discrete distribution and ditto density.

Table 22
Summary statistics for the scenario differences of our realistic 20-project portfolio

Diff	Min	1st qu	Med	Mean	3rd qu	Max
1 − 2	−69.3	7.2	17.7	16.6	29.2	97.2
1 − 3	−59.5	17.0	27.8	27.5	39.6	110.6
2 − 3	−71.2	−0.32	10.1	10.8	21.6	94.3

When the number of projects increases, the shape of the discrete distribution will approximate that of the continuous density. In Fig. 12 it is already possible to spot that trend, although at some points there are still zero-density spots. Another observation is that the distributions shift to the left when requirements creep and volatility combined with time compression are uniformly present in the management style. In our IT-investment portfolio this is reflected in its portfolio yield being lower with a higher probability. Note that in the business cases of each individual project we already discounted for higher cash-flows in case of creep and/or strict deadlines. But even then, the entire portfolio performance of this realistic IT-investment portfolio yields higher when the plans and schedules do not change (which is less realistic in practice). Of course, with intermediate scenarios we can optimize the added value further, and project higher yields. But then flexible management styles need to be part of the organizational culture. More information on recognizing management styles and their pitfalls can be found in another paper, where we analyze IT-project data to quantify the effects of certain IT-governance rules and styles [37].

The cost of mitigation. With the discrete densities, we can obtain an idea of the differences in added value between scenarios. This difference is an indicator of how much investment in mitigating the uncertainties and risks is acceptable. Moreover if you plan more such investments the cost of professionalizing the IT-function can be amortized over various programs. In this case the cost differences are more than enough to cater for generous mitigation measures to prevent project failure, unnecessary requirements creep and nonbeneficial time compression. In Table 22 we provide summary statistics of the differences between the various scenarios. For example, suppose we executed the IT-investment portfolio in scenario 1 and in scenario 2. Then the difference in portfolio yield between those scenarios is going to be larger than $7.2 mn in 75% of the cases (the first quartile) and in 50% of the cases (the median) the difference is going to be larger than 17.7 million dollars. This implicates that if less than this amount is spent on mitigating unnecessary requirements creep by organizational, managerial, and process improvements, there is a positive pay-off, and a future pay-off for other IT-investments.

When apart from scope creep, also senseless deadline pressure can be capped, the benefits are huge: in 75% of the cases this leads to an increased portfolio yield of 17 million and in 50% of the cases think of $27.8 million. So even very high costs of dealing with those uncertainties in IT-development are justified, and in this realistic IT-investment portfolio the direct benefits are tremendous. Apart from that the indirect benefits for future investments are a multiplier on the investment in a more professional IT-function, where change control, and correct scheduling in combination with proper capacity management are immersed in the organization.

An SEI benchmark on the cost of professionalizing the IT-function is 1–8% of the annual IT-budget. In another paper where we audit the added value of such an investment, 4–6% of the annual IT-budget was spent with a high yield [34]. So if part of the projected benefit of our 37 million portfolio is spent beforehand on such a program, its benefits will outweigh the added cost. Of course, supposing that the portfolio is part of an ongoing IT-development improving practice.

In Fig. 13, we have depicted density functions of the yield of the 20-project portfolio for all three scenarios. The first plot represents scenario 1, and the most non-zero density is around the 60–70 million dollars, but there is a long tail to the left indicating a relatively large chance that the portfolio yield can be halved. The probability that the yield of the portfolio dives under zero seems negligible for this scenario. In case we have not mitigated requirements creep, we note that indeed the PDF shifts to the left, and it seems that the density dip of scenario 1 around the 50 million is superpositioned somehow in scenario 2. We still see the halloween-shape on the left but the dip is gone. Non-zero density is now roughly between $25 and $55 million. So chances are lower that the highest possible yield is obtained with scenario 2. Then again, chances that the investment is going to be lossy are close to zero. Of course, the cash-flows of the business cases are the reason for that, and the business cases for all projects are quite healthy. Plot 3, representing scenario 3 is shifted most to the right, and the projected yields are indeed lower: roughly between 10 and 45 million. Still, chances on a lossy scenario seem close to nil.

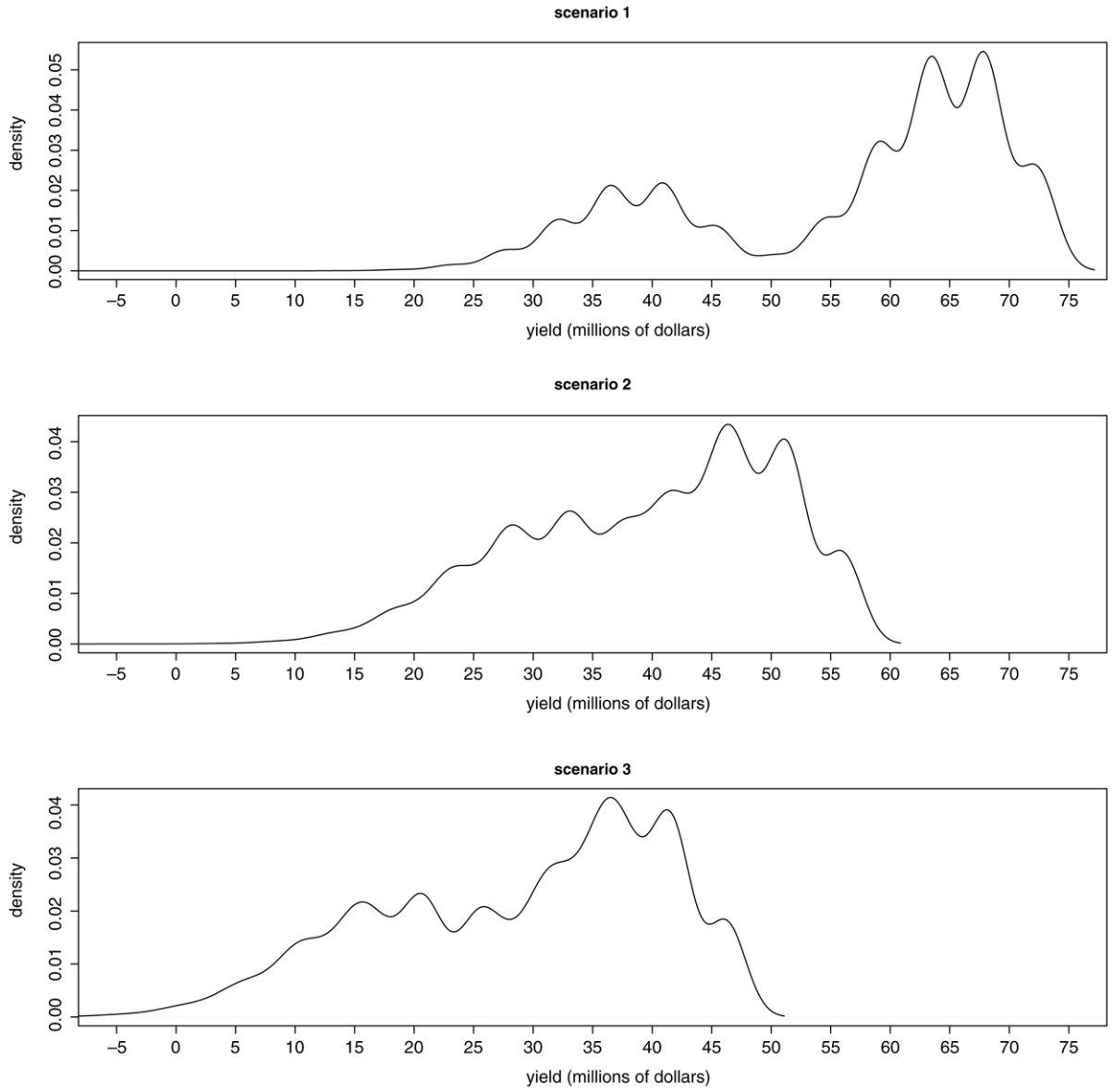

Fig. 13. Probability density functions of the portfolio yield of a 20-project portfolio for all three scenarios.

As said before, the shapes of the densities are interesting for comparisons and the search for theoretical densities underlying the portfolio yield distribution. But to connect probabilities to such a graph the approximation by using a histogram is a pragmatic alternative. In Fig. 14, we depict such a histogram for all the three scenarios. For instance, in scenario 1 the chance that the portfolio yield is between 60 and 62 million is about 20%, as we can spot directly from plot 1. The highest bar in plot 2 is around the 50 million, with a chance of occurrence of about 17%. Chances on a 60 million portfolio yield are zero in this scenario. In scenario 3 we spot that around the 40 million we have the highest bar with a probability of about 13%. So the plots show a lower top-yield with a lower probability from scenario 1 to 3.

In Fig. 15 and Table 23 we summarized the yield distribution of Fig. 12. As before we use the box and whiskers plot to obtain more insight into the distributional properties of the portfolio yield per scenario. The dots in all three plots are representing outliers. Indeed this shows that there are long tails to low and lossy yields. We know already that they have a low chance of materializing. In fact the probability that the portfolio yield is above zero is approximating to 1 for scenario 1. So that is really negligible. The scenario 1 box is high up, and from the PDF we know that

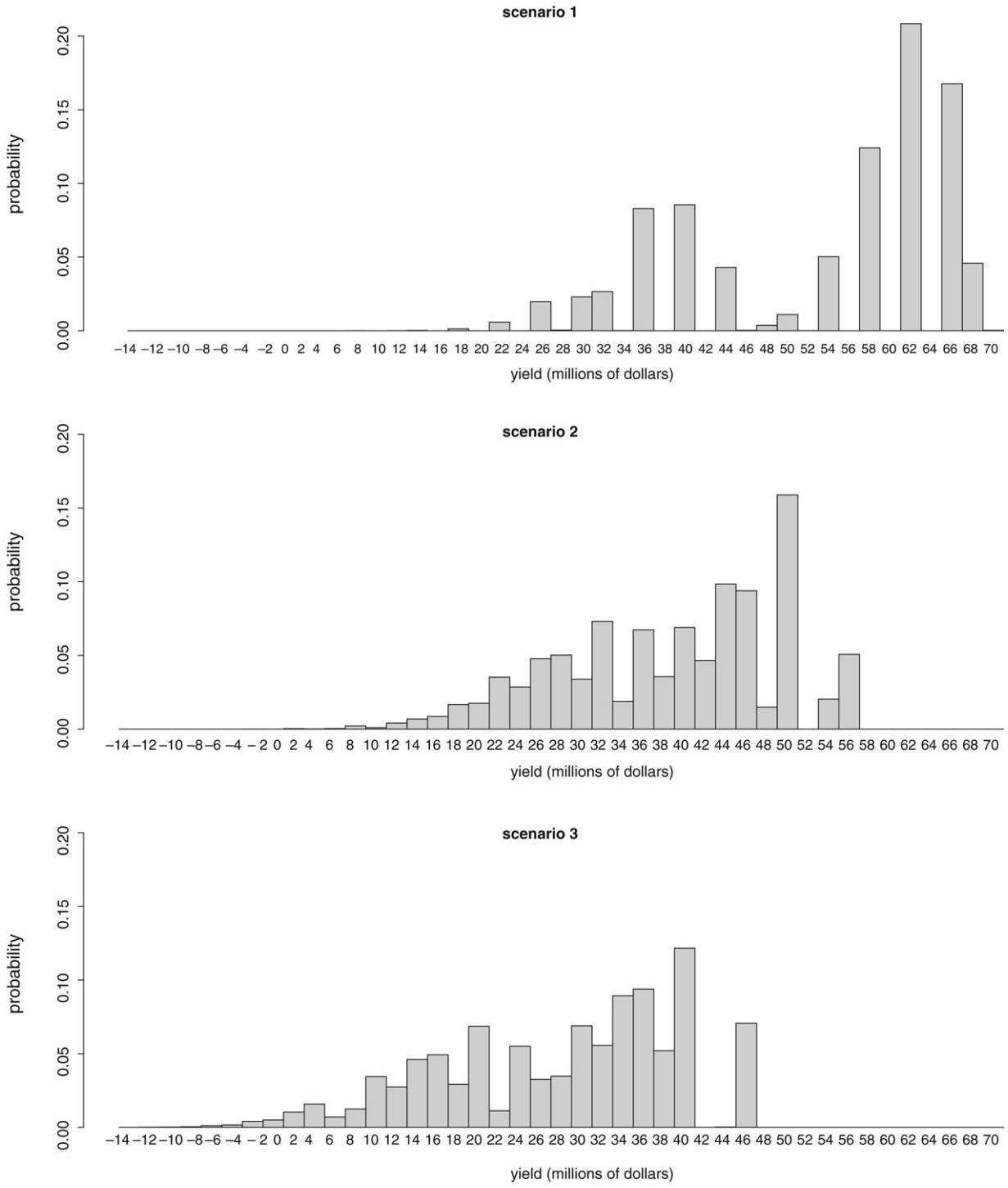

Fig. 14. Three histograms for the 20-project portfolio.

there are no significant zero-densities in the 50% range. So indeed the chance that the portfolio yield is between 45 and 67 million dollars is 50%. Those precise numbers are displayed in Table 23. Also the median is pretty high, implying that ending in the upper 50% of the yield distribution leads to a portfolio yield between 63 and 72 million dollars.

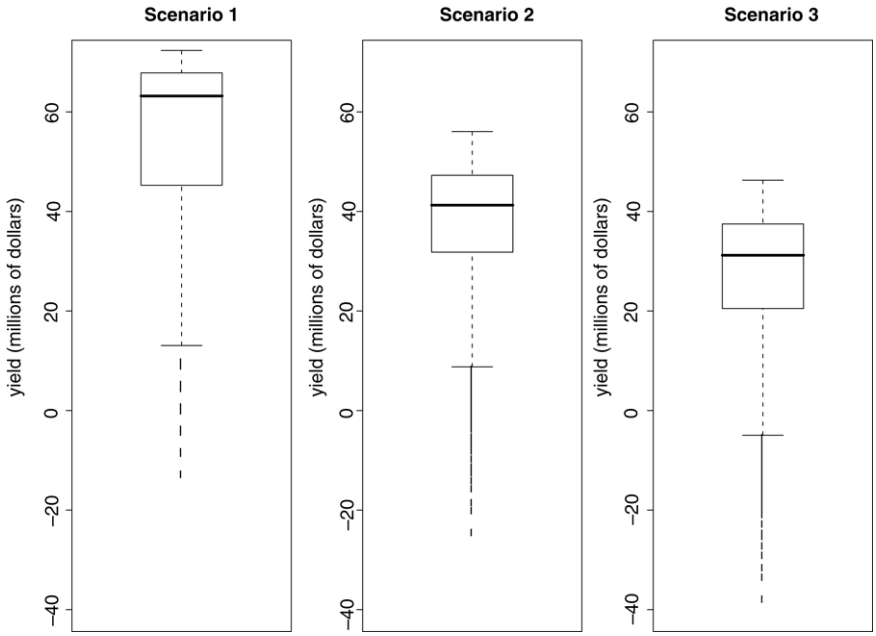

Fig. 15. Box plots of the yield distribution of a 20-project IT-portfolio for all three scenarios.

Table 23
Summary statistics for the three scenarios of our realistic 20-project portfolio

#	Min	1st qu	Med	Mean	3rd qu	Max
1	−13 226 349	45 261 578	63 198 303	56 568 931	67 835 496	72 339 175
2	−24 889 282	31 814 230	41 261 235	39 943 140	47 254 758	56 047 048
3	−38 280 985	20 502 537	31 195 745	29 095 986	37 492 808	46 285 098

Table 23 also summarizes the distributional properties corresponding to scenario 2. The box is lower than that of scenario 1, and the maximal yield is lower. It is 16 million lower, which is a lot on a maximal yield of 73 million for scenario 1. The median is also much lower, almost 20 million dollars. Still the chances on a lossy investment are close to zero: there is a 99.99387% chance that the portfolio yield is above zero. The probability that we end up between 32 and 47 million is 50%, and for the upper 50% we can reach between 41 and 56 million. Still a very favorable investment, but capping unnecessary requirement creep is going to pay-off considerably.

In scenario 3, we note from the plot and tabulated figures that the bulk of the data (the box) is uniformly below the 40 million, whereas in scenario 1 this is uniformly above 40 million. In fact there is a 50% chance that the portfolio yield is going to end up between 20 and 37 million dollars. On a reassuring note, the chances on a positive yield are huge: 99.22247%. Still the chances on the highest possible portfolio yields are zero compared to the other two scenarios, since in this scenario the uncertainties with respect to requirements creep combined with strict deadlines are materialized.

Managing this IT-portfolio in such a manner that only beneficial additional requirements are allowed and only paying-off time-to-market schedule pressure is supported will achieve much higher portfolio yields at lower risk. Needless to say that this kind of quantitative information on planned IT-investment portfolios is of large value in the process of decision making, its optimal implementation, and its prudent management. Of course, also other more qualitative aspects play an important role.

In Fig. 16 we depict the cumulative portfolio yield distributions for all scenarios. We put them in a single picture for easier comparison. The cumulative yield distribution provides much more information to the decision maker than just the knowledge of gain (in terms of a simple NPV), or the expected yield, although that one is already more informative than the net present value where risk of failure is excluded. From the cumulative probability distribution one can easily derive the probability of getting a lower yield than some fixed amount. Depending on the risk appetite – ranging from

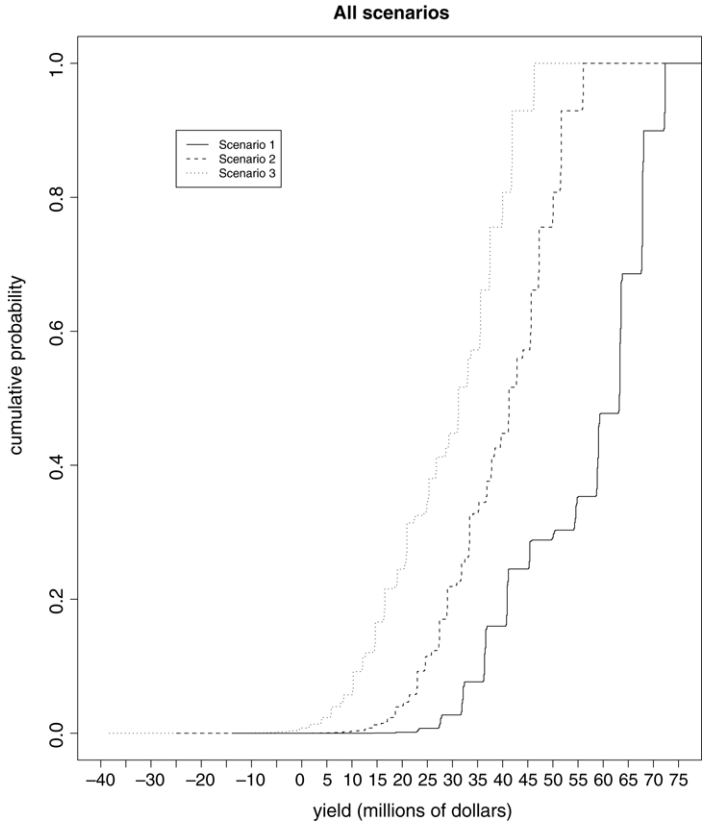

Fig. 16. Cumulative yield distributions of a 20-project IT-portfolio for all three scenarios.

risk averse to thrill seeking – an executive can now take a well-considered decision regarding acceptance of the proposed investment portfolio. Moreover, to maximize the yield, certain risk scenarios can be worked out. From this, and the cost avoidance by implementing risk mitigation plans one can calculate the portfolio yield in a much more realistic manner than by a simple discounted cash-flow analysis that does not take IT-specifics into account, and merely looks at the discount rates common for other investments with a much friendlier risk profile. In this graph the cumulative probability is on the vertical axis. For instance, 0.2 corresponds to a chance up to 20%. When we draw a horizontal line from that point we meet the three scenario lines: scenario 3 we intersect at first, represented by the dotted staircase. At the intersection we find the portfolio yield with a chance up to 20% of realizing, which is \$16.5 mn. For scenario 2 this is \$29 million and for scenario 3 this equals 41 million dollars. Or we can draw a vertical line, suppose we are interested to know what the chances are of obtaining a portfolio yield of more than 30 million dollars. Then we can easily infer that for scenario 1 this amounts to 97.3%, in case of scenario 2 this chance is much lower: 78.1%, and in case of scenario 3, the portfolio yield is going to be above 30 million dollars with a chance of 55.2%.

Furthermore, in Fig. 16 we can loosely interpolate other scenarios. Suppose that there is some requirements creep but not on every project. This rigid scenario with some creep will end up in Fig. 16, somewhere between the solid and dashed line, that border the entirely rigid scenario and the overall requirements creep one. This is not entirely precise, since we then have to assume that the PDFs are of the same family of distributions, and given their shape (see Fig. 13) it seems okay to do so in order to obtain an idea. Of course, when different management styles are applied to various parts of the IT-portfolio, this can also be made precise in a special intermediate scenario, and calculated analogously to the three scenarios. Similarly, an intermediate scenario where some parts are rigid, some have requirements creep and some have time compression as well, ends up somewhere between scenario 1 and 3. And if there is overall requirements creep and some useless deadline pressure we end up between 2 and 3. For large investments and preconceived plans on implementations, a special scenario analysis in addition to the three we showed here is recommended, so that for that case the probability distribution of the portfolio yield is derived.

Incidentally, Fig. 16 clearly displays how a great plan with a huge projected yield can utterly fail by mediocre implementation practice. Suppose that the zero-yield-line is at 55 million dollars: then a mostly rigid scenario pays-off when the risk of failure is also properly managed, but the other two scenarios are going to be lossy. In fact this is the portfolio equivalent of our fourth example project – the new payment system – of which we projected a positive NPV for the rigid scenario but a negative NPV for a scenario with growing demands during implementation without deadline extensions. For example, government projects fall prey to these scenarios where politicians set deadlines driven by law making and voters potential, rather than optimal yield for tax payers money. At the same time any form of change control is absent, which is corroborated by the high requirements creep benchmarks for civil and military software [23]. This deadly combination was in the cases that we investigated, failing government projects often the cause of a huge waste of public capital.

10. Risk-adjusted discount rates

In this section we address the question whether we would have found the same results if we had used a higher discount rate in our discounted cash calculations. From practice and in the literature the notion of Weighted Average Cost of Capital, or WACC, is a well-known concept. It is used as the discount rate in discounted cash calculations and is based on company-specific key figures, such as cost of equity, debt, market value and the corporate tax rate. In our examples we have used the common 12% as cost of capital. One could raise the WACC to account for the risk exposure in discounted cash calculations. Pisello and Strassmann [29, p. 41] propose to add to the normal cost of capital 0% for no-risk IT-investments, 10–15% for low-risk ones, 15–30% for medium risks, and for high risk IT-investments 30% or higher. In an earlier paper [35] we quantified the value of a single IT-project, and we adjusted the WACC as well. These methods are providing the decision maker with an idea of the viability of an IT-project and its yield. The method that we propose in this paper is more fine-grained and is better suited for calculating the portfolio yield in case of larger investments consisting of entire IT-portfolios.

10.1. The cost of IT

In [35] we introduced the concept WACIT, which stands for the Weighted Average Cost of Information Technology as the premium to increase the WACC in case of IT-risk exposure. To obtain a quick idea of the net present value, the following rule of thumb is sometimes used by us. Set the WACIT equal to the chance of failure using Formula (7) (or a similar one that is better suited) and increase this number with 10% if requirements creep or time compression play a role. If both requirements creep and time compression play a role then increase the chance of failure with 20%. By doing so an estimate of the lower bound of the net present is obtained. This is important, because in practice an NPV is often estimated much to be optimistic. Similar rules of thumb can be found in [29,35].

We now show what lower bound is found for the NPV of our example projects when we apply the WACIT-approach with the above rule of thumb. Let r_1 denote the risk-adjusted discount factor for project 1 with a requirements creep scenario. Let r_2 be the risk-adjusted discount factor for project 2 where we have a scenario with requirements creep and time compression. Furthermore, let r_3 be the risk-adjusted discount factor for project 3 with requirements creep, and denote r_4 to be the risk-adjusted discount factor for project 4 with requirements creep. Applying our rule of thumb yields the following values for the various discount rates.

$$\begin{aligned} r_1 &= 12 + 5.05 + 10 = 27.05\% \\ r_2 &= 12 + 13.02 + 10 + 10 = 45.02\% \\ r_3 &= 12 + 17.17 + 10 = 39.17\% \\ r_4 &= 12 + 28.47 + 10 = 50.47\%. \end{aligned}$$

Since the chance of failure is estimated by using Formula (7) the corresponding numbers are given with a precision of two digits behind the decimal point. The numbers 10 and 12 are constants, with infinite precision, they are not measured at some accuracy level. Therefore there is no need to round the resulting rates. Apart from that, for explanatory reasons, we keep the precision in, so that others can more easily repeat our calculations.

These discount rates are much higher than the 12% that you would normally use in discounted cash-flow analyses. We discounted for three sources of IT-risk by using a mix of benchmarks and experience-based premiums: the 10%

Table 24
The WACIT-based and expected yield compared

	Project 1	Project 2	Project 3	Project 4
r_i	27.05%	45.02%	39.17%	50.47%
WACIT(NPV)	\$60 724	\$1825 126	\$1 135 651	\$1312 728
$E(Y)$	\$61 629	\$2794 577	\$4416 891	\$1864 574

Table 25
The discount factors for the various discount rates that we used in this paper to calculate net present values

Rates	r	r_1	r_2	r_3	r_4
	12	27.05	39.17	45.02	50.47
t			discount factors		
1	0.893	0.787	0.719	0.690	0.665
2	0.797	0.620	0.516	0.475	0.442
3	0.712	0.488	0.340	0.328	0.294
4	0.636	0.384	0.233	0.226	0.195
5	0.567	0.302	0.160	0.156	0.130
6	0.507	0.238	0.110	0.108	0.086
7	0.452	0.187	0.076	0.074	0.057
8	0.404	0.147	0.052	0.051	0.038
9	0.361	0.116	0.036	0.035	0.025
10	0.322	0.091	0.025	0.024	0.017

additions. We accounted for the requirements creep risk and time compression exposure by adapting the important IT-project KPIs and the failure risk was accounted for by adjusting the net present value in case of loss at a certain probability.

In Table 24, we displayed the outcomes of the appraisals for all 4 projects, both for the WACIT-based approach and by deriving the portfolio yield distribution. Indeed the WACIT-based NPV-estimation produces a uniform lower bound for the expected yield for all projects in our sample portfolio. To support go/no-go decisions this first but rough indication is very useful for the decision maker, if it concerns a single IT-investment project and not much data is available. In case of an entire IT-portfolio our more accurate method which characterizes the stochastics of the portfolio yield is preferable. Indeed, for projects 1, 2 and 4 the lower bound is not that pessimistic. But for project 3 it is. The reason is that on the one hand, the requirements creep scenario has different cash-flows than in the WACIT-approach. So for the WACIT-NPV we used the cash-flows for the rigid scenario to simulate the situation where only limited information on a single scenario is present. On the other hand requirements creep is perceived to be an additional IT-risk – which it often is – but not in this case. This causes a much lower net value using the rule of thumb than you would conclude from its yield distribution. In practice, decision makers sometimes need answers very quickly, even if the answers will be less accurate. In this case, the WACIT-approach provides a very pessimistic lower bound, but nonetheless, the investment is still appraised to be very valuable. So, an executive will presumably engage in this portfolio, provided that other more qualitative aspects are also favorable. When more information becomes available, a more fine-grained calculation can be made. In our example case this is an adapted set of cash-flows due to new insights that strongly improve the original ideas. The ensuing expected project yield turns out to be a factor 4 higher, boosting the total portfolio yield.

In Table 25 we summarized the discount factors that we used in our NPV-appraisals. In fact, this table just displays the discount factors $1/(1 + r^t)$ used in the NPV-formula (1). Here, t stands for the year of investment, and r for the discount rate used. In the second row of Table 25 we provide those rates, ranging from the commonly used WACC of 12%, to the higher discount rates that take the WACIT into account as well.

11. Conclusion

In this paper we have presented a method to allow for IT-risks and uncertainties in calculating the value of a portfolio of IT-investment proposals. On the basis of minimal data an accurate and realistic picture can be made of

the gain of the IT-investments, measured in terms of the net present value, a common economic indicator. We use various scenarios in which common sources of IT-risk such as failure, cost/time overruns, and uncertainties such as requirements creep and time compression occur. For all kinds of scenarios we can calculate the best gain in terms of the net present value, indicating what management style to use for which project in an IT-investment portfolio. Moreover, we account for the risk of project failure within a portfolio by deriving the probability distribution of the portfolio yield. So we can produce all the relevant distributions and other visual aids, like box plots, histograms, and more. The portfolio yield is a random variable due to the risk regarding project delivery, and the uncertainty of requirements creep and time compression. With the aid of such distributions, a much richer information source is available to the decision maker who can then take a deliberate decision regarding portfolio acceptance under a presumed scenario based on a quantified insight in the risk exposure of the portfolio given a certain risk appetite of the organization.

Exactly the same approach can be applied to analyzing other economic indicators such as the return on investment, the pay back period, the internal rate of return, the risk-adjusted return on capital, and so on. All data needed for appraisals using such indicators is present in this paper. For details on how to apply these analogous techniques within the context of valuing of IT-investments we refer to [35], where for a lot of standard economic indicators an example is treated elaborately.

In this paper, we have kept our examples as simple as possible to illustrate the essence of our method. All assumptions have been made explicit, so that everybody can adjust the assumptions to their own specific situation. For example, in the case of specific investment premiums or statutory write-offs, discounted cash-flows can be adjusted without any problem. The quintessence of quantifying the value of a risk-bearing IT-investment portfolio does not change by that.

Acknowledgements

This research received partial support by the Dutch *Joint Academic and Commercial Quality Research & Development (Jacquard)* program on Software Engineering Research via contract 638.004.405 *Equity: Exploring Quantifiable Information Technology Yields* and contract 638.003.611 *Symbiosis: Synergy of managing business-IT-alignment, IT-sourcing and offshoring success in society*. Furthermore, we like to thank a number of organizations that will remain anonymous for discussions on the value of their IT-investment portfolios. Also thanks to our PhD students for their splendid support, in particular Laurenz Eveleens. Finally, we are grateful to George Tillman, former CIO of Booz, Allen & Hamilton, and Masafumi Kotani, IBM Asia Pacific and Japan, for their valuable comments on the first draft version of this paper. We thank George Tillman for scrutinizing our manuscript and going through all our computations again. His comments confirmed once again that this kind of research meets some of the urgent needs of the IT-industry. He found a few calculation errors that have been corrected. Masafumi Kotani has raised the interesting question whether our definition of uncertainty and risk is in line with the principles of the Real Option Value (ROV) approach, which is sometimes used for justifying (usually large) infrastructure investment proposals. In Section 2 we dive into this matter. Further we like to thank the anonymous reviewers for their encouraging words, and very good suggestions.

References

[1] A.J. Albrecht, Measuring application development productivity, in: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1979, pp. 83–92.
[2] A.J. Albrecht, J.E. Gaffney, Software function, source lines of code, and development effort prediction: a software science validation, IEEE Transactions on Software Engineering 9 (9) (1983) 639–648.
[3] P. Allen, S. Frost, Component-Based Development for Enterprise Systems, Cambridge University Press, 1998.
[4] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, Springer, 1993.
[5] J. Bloem, M. van Doorn, P. Mittal, Making IT Governance Work in a Sarbanes-Oxley World, John Wiley & sons, 2006.
[6] B. Boehm, B. Clark, C. Westland, R. Madachy, R. Selby, Cost Models for Future Software Life Cycle Processes: COCOMO 2.0, in: J.D. Arthur, S.M. Henry (Eds.), Annals of Software Engineering Special Volume on Software Process and Product Measurement, 1995.
[7] B.W. Boehm, C. Abts, A.W. Brown, S. Chulam, B.K. Clark, E. Horowitz, R. Modachy, D. Reifer, B. Steece, Software cost estimation with COCOMO II, Prentice-Hall, 2000 (last update September 2002).
[8] R.C. Clelland, J.S. deCani, Francis E. Brown, Basic Statistics with Business Applications, 2nd ed., John Wiley & Sons Inc., 1973.
[9] D. Dalcher, A. Genus, Introduction: Avoiding IS/IT implementation failure, Technology Analysis and Strategic Management 15 (4) (December 2003) 403–407.
[10] J.B. Dreger, Function Point Analysis, Prentice Hall, 1989.

[11] A.W. Brown (Ed.), Component-Based Software Engineering, IEEE Computer Society Press, 1996.
[12] V. Basili, et al., Lessons-Learned Repository for COTS-BASED SW Development, the DoD Software Tech News 5 (3) (2002) 4–7. 20–21.
[13] D. Garmus, D. Herron, Function Point Analysis — Measurement Practices for Successful Software Projects, Addison-Wesley, 2001.
[14] International Function Point Users Group, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting practices manual, Technical Report ISO/IEC 20926:2003, International Standardization Organization, www.iso.org November 2003.
[15] META Group, The Business of IT Portfolio Management: Balancing Risk, Innovation, and ROI, Technical report, META Group, Stamford, CT, USA, January 2002.
[16] The Standish Group, CHAOS, 1995, Retrievable via: standishgroup.com/visitor/chaos.htm (Current February 2001).
[17] The Standish Group. CHAOS: A recipe for success, 1999, Retrievable via: www.pm2go.com/sample_research/chaos1998.pdf.
[18] The Standish Group, EXTREME CHAOS, 2001. Purchase via: https://secure.standishgroup.com/reports/reports.php.
[19] R.V. Hogg, J.W. McKean, A.T. Graig, Introduction to Mathematical Statistics, 6th ed., Prentice Hall, 2005.
[20] C. Jones, Assessment and Control of Software Risks, Prentice-Hall, 1994.
[21] C. Jones, Applied Software Measurement: Assuring Productivity and Quality, 2nd ed., McGraw-Hill, 1996.
[22] C. Jones, Estimating Software Costs, McGraw-Hill, 1998.
[23] C. Jones, Software Assessments, Benchmarks, and Best Practices, in: Information Technology Series, Addison-Wesley, 2000.
[24] E. Jordan, L. Silcock, Beating IT Risks, Wiley & Sons, 2005.
[25] C.F. Kemerer, Reliability of function points measurement – a field experiment, Communications of the ACM 36 (2) (1993) 85–97.
[26] C.F. Kemerer, B.S. Porter, Improving the reliability of function point measurement: An empirical study, IEEE Transactions on Software Engineering SE-18 (11) (1992) 1011–1024.
[27] F. Mosteller, J.W. Tukey, Data Reduction and Regression, Addison-Wesley, 1977.
[28] M.C. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley Publishing Company, Reading, MA, 1995.
[29] T. Pisello, P.A. Strassmann, IT value chain management – maximizing the ROI from IT investments, in: volume D10 of Digital Publications from The Information Economics Press, The Information Economics Press, New Canaan, Connecticut, USA, 2003.
[30] L.H. Putnam, W. Myers, Measures for Excellence – Reliable Software on Time, Within Budget, Yourdon Press Computing Series, 1992.
[31] L.H. Putnam, D.T. Putnam, A data verification of the software fourth power trade-off law, in: Proceedings of the International Society of Parametric Analysts – Sixth Annual Conference, vol. III(I), 1984, pp. 443–471.
[32] J.W. Tukey, Exploratory Data Analysis, Addison-Wesley, 1977.
[33] C. Verhoef, Quantitative IT Portfolio Management, Science of Computer Programming 45 (1) (2002) 1–96. Available via: www.cs.vu.nl/~x/ipm/ipm.pdf.
[34] C. Verhoef, Quantifying Software Process Improvement, 2004. Available via: www.cs.vu.nl/~x/spi/spi.pdf.
[35] C. Verhoef, Quantifying the Value of IT-investments, Science of Computer Programming 56 (3) (2005). Available via: www.cs.vu.nl/~x/val/val.pdf.
[36] C. Verhoef, Quantitative Aspects of Outsourcing Deals, Science of Computer Programming 56 (3) (2005). Available via: www.cs.vu.nl/~x/out/out.pdf.
[37] C. Verhoef, Quantifying the effects of IT-governance rules, Science of Computer Programming 67 (2–3) (2007) 247–277. Available via: www.cs.vu.nl/~x/gov/gov.pdf.
[38] E. Yourdon, Death March — The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects, Prentice-Hall, 1997.